On Protocols for Information Security Services

Debasish Jena

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

On Protocols for Information Security Services

Dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Debasish Jena

(Roll- 50706001)

under the guidance of

Prof. Sanjay Kumar Jena

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Orissa, 769 008, India

2010

dedicated to my father...

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India.

Dr. Sanjay Kumar Jena
Professor

April 5, 2010

Certificate

This is to certify that the work in the thesis entitled ***On Protocols for Information Security Services*** by ***Debasish Jena*** is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Computer Science and Engineering in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Sanjay Kumar Jena

Acknowledgement

This dissertation, though an individual work, has benefited in various ways from several people. Whilst it would be simple to name them all, it would not be easy to thank them enough.

The enthusiastic guidance and support of *Prof. Sanjay Kumar Jena* inspired me to stretch beyond my limits. His profound insight has guided my thinking to improve the final product. My solemnest gratefulness to him.

I am also grateful to *Prof. Shakti Ranjan Mohapatra, Prof. Gopal Krishna Nayak, Prof. Ajit Kumar Das* and *Prof. Sudarsan Padhy* for their ceaseless support throughout my research work. My sincere thanks to *Prof. Banshidhar Majhi* for his continuous encouragement and invaluable advice.

It is indeed a privilege to be associated with people like *Prof. S. K. Rath*,*Prof. R. Baliarsingh*, *Prof. D. P. Mohapatra*, *Prof. A. K. Turuk* and *Prof. B. D. Sahoo*. They have made available their support in a number of ways. My humble acknowledgement to *Prof. S. K. Patra*, and *Prof. G. K. Panda* for nourishing my intellectual maturity.

Many thanks to my comrades and fellow research colleagues. It gives me a sense of happiness to be with you all. Special thanks to *Saroj*, whose involvement gave a new breath to my research.

Finally, my heartfelt thanks to my wife *Pranati* and my children *Guddy, Sibu*, and *Alpha* for their unconditional love and support. Words fail me to express my gratitude to my beloved parents and parents-in-law who sacrificed their comfort for my betterment.

Debasish Jena

Abstract

Now-a-days, organizations are becoming more and more dependent on their information systems due to the availability of high technology environment. Information is also treated as vital like other important assets of an organization. Thus, we require Information Security Services (ISS) protocols to protect this commodity. In this thesis, investigations have been made to protect information by developing some ISS protocols.

We proposed a key management protocol, which stores one-way hash of the password at the server, instead of storing plaintext version of password. Every host and server agrees upon family of commutative one-way hash functions. Due to this prevention mechanism, online and offline guessing attacks are defeated. The protocol provides host authentication. As a result, man-in-the-middle attack is averted. It also withstands malicious insider attack.

Symmetric and asymmetric are two categories of cryptosystems, which are used to encrypt messages. Asymmetric cryptosystem for large message encryption incurs more computational overhead. Traditionally, asymmetric cryptosystems used to transfer the shared key, and then the symmetric cryptosystem is used to encrypt large messages. That's why, in this thesis, a unique asymmetric cryptosystem for encrypting large messages has been proposed, which is not only efficient but also secure as compared to other asymmetric cryptosystems. In consideration to all the aspect of efficiency and computation, our proposed scheme uses elliptic curve cryptosystem.

Blind Signature Schemes (BSS) facilitate a requester to obtain signature from a signer on any document, in such a way that the signer can't know anything about the message that is being signed. Four BSS have been proposed which are based on Elliptic Curve Discrete Logarithm Problem (ECDLP). These schemes utilize the intrinsic advantage of ECDLP in terms of smaller key size and lower computational overhead to its counterpart public cryptosystems such as RSA and ElGamal. The correctness, untraceability, blindness and unforgeability properties of the schemes have been proved. The proposed schemes are implemented and compared with known BSS. From the comparison, it is found that the proposed

schemes outperform all known previous algorithms based on IFP and DLP.

A remote user authentication scheme is a two-party protocol whereby an authentication server confirms the identity of a remote individual logging on to the server over a non-trusted and unsecured network. Password based authentication schemes are commonly used for authenticating remote users. Two unique and efficient remote user authentication scheme using smart cards based on ECDLP have been proposed. The proposed schemes do not require verifier table and allows the user to choose their passwords. As the schemes are based on ECDLP, they require less bit key size and less computation than their counterpart based on IFP or DLP. Because of these properties, they can be easily implemented in smart card. They also withstands message replaying attack.

***Keywords*:** Information Security Services, Elliptic Curve Discrete Logarithm Problem, Large Message Encryption, Blind Signature, Remote User Authentication.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 1

Introduction

If we look at to the history of Information Security (IS), well back to the World War II, there was need to secure physical location, hardware and software from outside threats. Multiple levels of security were proposed and implemented to protect information and data integrity. Access to sensitive military location was controlled through use of keys, identity card and facial recognition of authorized personnel by the security guard. During those earlier years, IS was a straight forward process, comprising of physical security and simple document classification scheme. The primary concern to security was physical theft of equipment, sabotage or espionage against the product of the system. But with the progress of Information Technology, the IS issues have become much more complicated and gone beyond national and geographical boundaries [3, 4].

Over these period, an elaborate set of protocols and mechanisms has been developed to deal with IS Services (ISS) issues, when information is represented in electronic medium [5]. Information has not changed dramatically, but the storage and transmission of information has changed a lot. Thereby, it enables the users to copy and alter the information very easily, that cannot be distinguished from the original; sometime creates havoc in business, government and society at large.

The information security services are *Confidentiality*, *Integrity*, *Availability*, *Non-Repudiation*, *Authentication*, etc. [2, 6]. Confidentiality property is achieved through encryption. As the security of encryption lies mainly in its keys, not only do they have to be of sufficient length, but they also have to be shared securely. Shared secret keys have to remain secret.

Asymmetric cryptography was invented to address the drawbacks of symmetric cryptosystems, i.e., key management. The most obvious application of a public key encryption system is confidentiality; a user can communicate securely over a public channel without having to agree upon a shared key beforehand. But due to more overhead of asymmetric cryptosystems, traditionally, the symmetric cryptosystem is used to encrypt large messages. Before communicating, the key should be available at both sender and receiver end. So, to encrypt large messages, we usually take the help of both symmetric and asymmetric cryptosystems. Using asymmetric key cryptosystem, the key is exchanged, and the symmetric key cryptosystem is used to encrypt the large message.

One of the fundamental tool of ISS is the signature [7]. It is the building block for many services such as non-repudiation [8], authentication, integrity, identification etc. In case of non-repudiation, the requester and the service providers can be prohibited of denying the action made on the transaction made between them. In this purpose, digital signature scheme is used, which provides a way for signer to sign document electronically in a secure and efficient manner using his private key so that, the signatures can later be verified by anyone else by using public key of signer. We have studied digital signature and in particular we focus on variant of digital signature, i.e., the blind digital signature. The blind signature scheme (BSS) is used in application like Internet voting and Digital Cash, where the requester needs to get the signature in the message from the signer without really exposing the message content to the signer.

Password based remote user authentication schemes are used to check the validity of a login request made by a remote user to gain the access rights on an authentication server (AS). In these schemes, the AS and the remote user share a secret, which is often called as password. The remote user uses the password to create a valid login request to the AS. To provide the access rights to the user, AS checks the validity of the login request.

The remainder of this chapter is organized as follows. Existing literatures are reviewed in Section 1.1. The motivation for the thesis is formally stated in Section 1.2. Finally, the Section 1.3 outlines the organization of the thesis.

1.1 Literature Survey

To achieve the issues of ISS, the protocols in following areas are required:

- Key Agreement Protocol— for sharing shared secret key among the sender and receiver in cryptosystems.
- Encrypting large message using asymmetric cryptosystem which should take less computation time than the existing asymmetric cryptosystem.
- Digital Signature is the fundamental tool of ISS. Hence, it is required to design efficient digital signature protocols and in particular BSS for anonymity of the requester.
- Remote user authentication is used to address the authentication service of information security system.

The literature survey of above areas are discussed as follows.

1.1.1 Key Management Protocols

Key exchange protocols can be categorized into two party key agreement protocols (2PKAP) and three party key agreement protocol (3PKAP).

Two Party Key Agreement Protocols

The Diffie-Hellman key agreement protocol (also called exponential key agreement) was developed by Diffie and Hellman [9] in 1976. The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets. A number of commercial products employ this key exchange technique. However, this protocol does not authenticate the participants engaging in exchanging their session keys. This gives chance to an adversary to impersonate one of the participants. In 2000, Jean-francois Raymond et al. [10] shown that, Diffie- Hellman key agreement protocol implementations have been plagued by serious security flaws. The attacks can be very subtle. In 2004, Jiang et al. [11] proposed a key exchange protocol for set-top box (STB) and smart card based on Schnorr's digital signature protocol and one-way hash function. However, in 2006, Yoon et al. [12]

shown that protocol proposed by Jiang et al. [11] is vulnerable to an impersonation attack and does not provide perfect forward secrecy and proposed a new secure key exchange protocol based on a one-way hash function and Diffie-Hellman key exchange algorithm for secure communication between STB and Smart Card in Internet Protocol Television broadcasting. Later, Lee et al. [13] proved that Yoon et al.'s protocol is vulnerable to impersonation attack and can not achieve mutual authentication. In 2009, Xiumei et al. [14] also shown that protocol proposed by Yoon et al. [12] is susceptible to password-compromise impersonation attack and server compromise attack.

Password based protocols for authenticated key exchange was first suggested in 1992, by Bellovin et al. [15]. They claimed that their protocols are secure against active attacks and have the property that the password is protected against online dictionary attacks. Password based protocol has become quite popular and many researchers have proposed and applied to many communication systems [16–19]. Some of them only provide heuristic security analysis [15–18,20–26] and some have been formally proven secure [19, 27–29]. In 2002, Yeh et al. [30] have proposed a protocol called Simple Authentication Key Agreement Protocol (SAKA) which is simple and cost effective as compared to previously known protocols. They claim that their protocol is secure against both passive and active adversaries.

Three Party Key Agreement Protocols

2PKAP is not a particularly useful application, because they require $n(n-1)/2$ number of keys. A much more common scenario is that of three-party key agreement [31]. The model for three party key distribution is that two parties having no shared secret key, enlist the assistance of a mutually trusted third party which performs the actual key distribution. This trusted third party is frequently referred to as Authentication Server or Key Distribution Center (KDC). Each of the two parties is assumed to share a long term key with the AS. As with 2PKAP, the goal is to design a secure 3PKDP. The conditions for a secure 3PKAP are essentially similar to that of 2PKAP. The only additional requirement is that a 3PKAP must be secure against a malicious insider, i.e., a legitimate party that, by participating in legitimate runs of the protocol, can gather enough information

to impersonate other parties or otherwise abuse the protocol (e.g., a malicious insider disclosing a key shared with another party).

Steiner et al. (STW) [23] proposed a three party encrypted key exchange (EKE) protocol. Since this is a three party key agreement protocol, both the hosts share a secret key only with trusted third party. Ding et al. [20] have proved that this protocol is vulnerable to undetectable online guessing attacks. According to Chun-Li Lin et al. [26], this protocol is also vulnerable to offline guessing attacks. An attacker attempts to use a guessed password in an online transaction. Host verifies the correctness of his guess using responses from server. If his guess fails he must start a new transaction with server using another guessed password. A failed guess cannot be detected and logged by server, as server is not able to depart an honest request from a malicious request. In offline guessing attacks an attacker guesses a password and verifies his guess offline. No participation of server is required, so server does not notice the attack. If his guess fails, the attacker tries again with another password, until he finds the proper one. Among these classes of attacks, the offline password guessing attack is the most comfortable and promising one for an attacker. It is not noticeable and has no communication cost. Storing a plain text version of the shared password at the server is a constraint that cannot (or ought not) always be met. In particular, consider the problem of a user logging in to a computer that does not rely on a secure key server for authentication. It is inadvisable for most hosts to store passwords in either plain form or in a reversibly encrypted form. Chun-Li Lin et al. (LSH) [25] proposed a three party EKE. This protocol is secure against both the offline guessing attack and undetectable online guessing attacks but also satisfies the security properties of perfect forward secrecy. The most important requirement to prevent undetectable online guessing attacks is to provide authentication of host to server. In STW, there is no verifiable information for server to authenticate host. On the contrary, if there is any verifiable information for server combined with password will result in offline guessing attacks. LSH uses server public keys for this purpose. But this is not a satisfactory solution all the times and is impractical for some environments. Communication parties have to obtain and verify the public key of the server, a task which puts a high burden on the user. In fact, key distribution

services without public-keys are quite often superior in practice than public key infrastructure (PKI) [26].

1.1.2 Large Message Encryption Protocols

Symmetric cryptosystems, such as Twofish [32], Serpent [33], DES [34, 35], AES [36], Blowfish, CAST5, RC4, TDES, and IDEA [37] use identical cryptographic keys for both encryption and decryption. In order to ensure secure communications between everyone in a population of n people a total of $n(n-1)/2$ keys are needed, which is the total number of possible communication channels.

To overcome the problem of key management in symmetric cryptosystem, in 1976 Public-key cryptosystem has been invented by Diffie and Hellman [9]. Since then, numerous public-key cryptographic systems have been proposed. Their security is based on the difficulty of solving a mathematical problem. Over the years, many of the proposed public-key cryptographic systems have been broken and many others have been demonstrated to be impractical. Today, only three types of systems are considered both secure and efficient. Examples of such systems and the mathematical problems are:

- **Integer factorization problem (IFP):** RSA [38] and Rabin-Williams [37].
- **Discrete logarithm problem (DLP):** Digital Signature Algorithm (DSA) [39], the Diffie-Hellman key agreement scheme [9], the ElGamal encryption and signature schemes [40], the Schnorr signature scheme [41], and the Nyberg-Rueppel signature scheme [42].
- **Elliptic curve discrete logarithm problem (ECDLP):** the elliptic curve analog of DSA (ECDSA) [43, 44], Diffie-Hellman key agreement scheme [45], the ElGamal encryption and signature schemes [46] and the Nyberg-Rueppel signature scheme [47].

Besides these, other cryptosystems, e.g., visual cryptography [48], encryption using genetics and image patterns [49], Water marking for digital rights management [50], secure mobile agent communication [51] are also being used in specific applications. As per NIST, ECDLP is computationally harder problem than its

counterparts like DLP, IFP [1]. Figure 1.1 compares the time required to solve an instance of the ECDLP with the time required to solve instances of the IFP or DLP for various modulus sizes and using the best general algorithms known [1].

The running time is computed in MIPS years. As a benchmark, it is generally accepted that 10^{12} MIPS years represents reasonable security at this time [1]. In Figure 1.1, the time to break RSA and DSA are grouped together because the best algorithms known for IFP and DLP have approximately the same asymptotic running times. From Figure 1.1, we see that, to achieve reasonable security, RSA and DSA should employ 1024-bit modulus; while a 160-bit modulus is sufficient for elliptic curve cryptography (ECC). Moreover, the security gap between the systems increases dramatically as the moduli sizes increases. For example, 300-bit ECC is dramatically more secure than 2048-bit RSA or DSA.

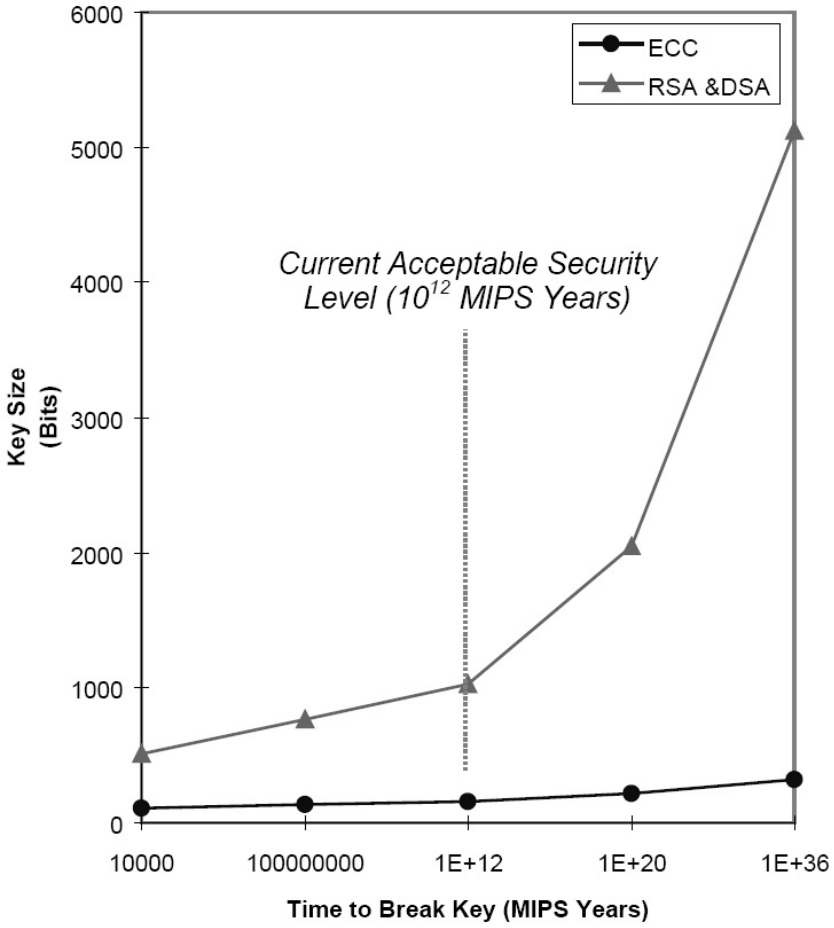


Figure 1.1: Comparison of Security Levels [1]

As asymmetric encryptions are more expensive than symmetric encryptions

such as DES [34, 35] and AES [36] in terms of computational cost, generally they are not directly used to encrypt large messages. Traditionally, if there is a need to encrypt a large message using an asymmetric cryptosystem, then a symmetric cryptosystem is used in addition. The message itself is encrypted using the symmetric cryptosystem and the symmetric key is encrypted using the asymmetric cryptosystem. To eliminate this requirement for an additional cryptosystem, in 2002, Hwang et al. [52] proposed a protocol to encrypt the long message using DLP. Later in 2004, Yuh-Dauh Lyuu et al. [53] in their cryptanalysis proved that Hawang et al.'s protocol is insecure.

1.1.3 Blind and Digital Signature Schemes

In 1978, Rivest et al. [38] proposed a digital signature protocol based on the factorization problem of number theory. The formal definitions of security for digital signatures were first outlined by Goldwasser et al. [54].

An interesting variant on the basic digital signature is the BSS. The concept of a Blind Digital Signature was introduced by Chaum [55] to enable spender anonymity in Electronic Cash Systems (ECS).

Any BSS must satisfy the following properties [55–58]:

- **Correctness:** The correctness of the signature of a message signed through the signature scheme can be checked by anyone using the signer's public key.
- **Unforgeability:** Only the signer can give a valid signature for the associated message.
- **Blindness:** The content of the message should be blind to the signer; the signer of the blind signature does not see the content of the message.
- **Untraceability:** The signer of the blind signature is unable to link the message-signature pair even when the signature has been revealed to the public.

Subsequently, many researchers suggested blind signatures based on IFP, Quadratic Residues (QR), DLP or ECDLP. Based on these schemes, researchers applied this in Digital Cash [59–62] and e-Voting [63–67]. An IFP blind signature scheme

based on the RSA digital signature scheme was proposed in 1983, by Chaum [68]. Hwang et al. [69] claimed that Chaum's scheme could not achieve untraceability and proposed an untraceable blind signature scheme based on the RSA cryptosystem to overcome the shortcoming [70].

A new blind signature scheme based on the RSA cryptosystem was proposed by Chaum [71] in 1987, which allows an unlimited number of signature types with only a fixed amount of computation. The scheme is very practical in some applications such as anonymous payment systems. Solms et al. [72] introduced the notion of perfect blackmail and money laundering in 1992. Then, Micali [73] introduced the concept of fair cryptosystems to prevent the misuse of strong cryptographic systems by criminals in 1993. However, Stadler et al. [74] considered that the anonymity and untraceability property could still possibly be misused by criminals. Hence, perfect blackmailing or money laundering would exist in places like anonymous payment systems. Stadler et al. suggested that a third trusted party, e.g., a Judge should be considered in the anonymous payment systems, to prevent such criminal acts. But, Hwang et al. [75] pointed out that the above blind signature scheme could in fact be traced by the signer. Coron et al. [76] informed that a signature forgery strategy, which is a branch of the chosen-message attack, might be introduced into the RSA digital signature system and cause trouble. Thus, a blind signature scheme was proposed by Fan et al. [56] to enhance the randomization of Chaum's blind signature scheme such that attackers cannot decipher what the signer exactly signs so as to avoid threats from chosen-message attacks. Unluckily, Hwang et al. pointed out that Fan et al.'s blind signature scheme could in fact be traced by the signer [77]. Chien et al. [78] proposed a partially blind signature scheme based on the RSA cryptosystem in 2001, which could reduce the size of the database and avoid double spending of the ECS. But, Hwang et al. [79] proved that the scheme of Chien et al. failed to meet the requirement of untraceability.

Based on QR [37], the following schemes have been proposed. Fan et al. [80] proposed a blind signature scheme in 1996. The security of the scheme is dependent on the difficulty of solving the square roots of QR without trapdoors. Later, Fan et al. [81] proposed a partially blind signature scheme that could reduce the

computation load and the size of the database for ECS in 1998. Later, Hwang et al. [69] proved that the scheme could not meet the requirement of untraceability that an ideal blind signature scheme should have. In the same year, Fan et al. [82] also proposed another blind signature scheme to further improve the computational efficiency. Then, Shao [58] claimed that Fan et al.'s blind signature scheme does not meet the requirement of untraceability and he proposed an improved user efficient blind signature of similar efficiency in 2000. Again, Fan et al. [83] did not only disagree with Shao's comments, but also presented a way to forge a legitimate signature so that the message could be signed by an attacker instead of the legal signer in Shao's blind signature scheme in 2001.

Based on DLP, the following schemes have been proposed. In 1994, Camenisch et al. [84] proposed two blind signature schemes. The first scheme was derived from a variation of DSA [85], and the second scheme was dependent on the Nyberg-Rueppel signature scheme [42]. Later, Harn [86] showed that, these two schemes could not achieve the property of untraceability in 1995. But, Horster et al. [87] claimed that Harn's cryptanalysis is not correct. When the signer traces the signature, he will obtain two pairs of signed messages that were satisfied by the equation of Harn's cryptanalysis. Therefore, the signer cannot trace back to the owner of the signature. Subsequently, Mohammed et al. [88] proposed a blind signature scheme based on the ElGamal digital signature scheme in 2000. However, in 2001, Hwang et al. [89] pointed out that this scheme could not achieve the property of correctness. When the requester obtained the blind signature from the signer, he/she could not unblind it to acquire the signature. In 2003, Lee et al. [90] asserted that Camenisch et al.'s schemes cannot satisfy the untraceability property of blind signature scheme. Subsequently, Lin-Chuan Wu [91] analyzed that Lee et al.'s traceability attack is flawed.

With respect to ECDLP, very few works have been done. Zuowen Tan et al. [92] have suggested digital proxy blind signature schemes based on DLP and ECDLP.

1.1.4 Remote User Authentication Protocols

Remote User Authentication scheme allows an authenticated user to access the remote server for accessing the services offered. Authentication is the key for information security; if the authentication mechanism is compromised, the rest of the security measures will be compromised.

In 1981, Lamport [93] introduced the first well-known hash-based password authentication scheme. In his scheme, the AS stores a verifiable table at the server to check the validity of the login request made by the user. However, high hash overhead and the necessity for password resetting is required. Thus, it is not practically implementable. In 1995, Haller [94] developed a prototype software system, the S/KEY™one-time password system, to defend, attacker from obtaining login id's and passwords of legitimate users. In 1995, Atkinson et al. [95] developed Onetime Passwords In Everything (OPIE) at U.S. Naval Research Laboratory which is an enhancement of Bellcore's S/KEY™1.0 package. OPIE improves on S/KEY™in several areas, including FTP service with one-time passwords, and a stronger algorithm for generating one-time passwords. In 1996, Mitchell et al. [96] shown that the Lamport scheme is vulnerable to a small n attack. In 1998, Shimizu et al. [97] proposed a password authentication method called PERM for application to e-mail forwarding. In 2000, Peyravian et al. [98] proposed a simple but efficient password authentication system. Their scheme is based on the collision-resistant hash function, such as SHA-1. Later in 2002, C. C. Lee et al. [99] shown that, in Peyravian et al. protocol, an attacker can easily obtain a user's password by guessing attack and then impersonate the user to login and access resources in the server. To overcome the vulnerability of their scheme, they proposed an improved scheme to enhance security of Peyravian et al. scheme. Again in 2008, Ji-Hye Park et al. [100] proposed secure password-based protocols for remote user authentication, password change, and session key establishment over insecure networks, which is an improvement of Peyravian et al. scheme. All the above protocols have a common feature, i.e., a verification password table should be securely stored in the AS. But, this property is a disadvantage for the security point of view. If the password table is stolen, removed or modified by the

adversary, the AS will be partially or totally broken/ affected. To overcome this problem, researchers published user authentication protocols which does not use verifiable table. Hence there is no threat of stolen of verifiable table.

In 1992, C. C. Chang [101] introduced the remote password authentication scheme with smart cards. In 1999, Yang et al. [102] proposed a timestamp-based password authentication scheme using smart card. In his scheme, users are permitted to choose and change their passwords freely and the remote server is not required to maintain a password table or verification table. However, in 2002, Chan et al. [103] pointed out the forged login attacks to the scheme is possible. In 2000, Sun et al. [104] proposed an efficient and practical remote user authentication scheme using smart cards. They claim that, their scheme significantly reduces the communication and computation costs. In the same year, Hwang et al. [105] proposed a scheme which is based on the ElGamal's public key cryptosystem. Their scheme does not require a system to maintain a password table for verifying the legitimacy of the login users. Based on simple geometric properties on the Euclidean plane, Wu [106] proposed an efficient smart card-oriented remote login authentication scheme in 1995. Like other smart card-oriented scheme, there is no verification table for authentication in Wu's scheme. Furthermore, the scheme allows a user to freely choose his password and requires much less computation overhead than other schemes. Wu's scheme is very efficient because it cleverly exploits the simple geometry property, but it also implicitly left a clue in the transmission messages. From the eavesdropped messages, Hwang [107] found that an illegal user can easily forge a valid message. So, the scheme is insecure. Wu's scheme shows that if the attacker can derive any geometric clue then, the attacker can break the system easily. Hwang did not show this key point, and did not propose his improvement.

Smart card based user authentication protocols overcome the problem of maintaining the verifiable table in case of password based user authentication protocol, but they require more computation time and more bit key as they use IFP or DLP. Hence they are not suitable for implementing in Smart Card as they have less computing power and memory.

1.2 Motivation of the Thesis

In this thesis, we have developed protocols for solving five problems which are encountered in Information Security Services. We describe below the motivation behind these problems.

1. As it has been mentioned earlier, one of the requirements for Information Security Services is efficient key management. To address this problem several researchers such as Diffie and Hellman [9], Raymond et al. [10], Jiang et al. [11], Yoon et al. [12], and Lee et al. [13] have developed many two party key agreement protocol. However, a two party key agreement protocol requires $n \times (n-1)/2$ keys, where n is the number of users. So when number of users becomes large, to manage large number of keys is a complex problem. To overcome this problem researchers like Steiner et al. [23], Ding et al. [20], and Chun-Li-Lin et al [25] have developed three party key agreement protocols. However, the work of Steiner et al. [23] suffers from undetectable online guessing attack and offline guessing attack while the work of Chun-Li-Lin et al. [26] uses expensive public key infrastructure(PKI). **Motivated by this, we propose to develop a protocol which will retain the advantages of earlier work and avoid attacks like online guessing attack, offline guessing attack and will not use the expensive PKI**.

2. For encryption of large messages, authors such as Schneier et al. [32], Biham et al. [33], Smid et al. [35] used symmetric key cryptography. It is well known that symmetric key cryptosystem is computationally faster but managing secret key is a complex problem. Use of public key cryptosystem achieves a higher level of security and avoids key management at the cost of more computational time as per researchers like Rivest et al. [38], ElGamal [40]. Hence, for encrypting large message, the message itself is encrypted using the symmetric key cryptosystem and the symmetric key is encrypted using asymmetric key cryptosystem. To eliminate the requirement for an additional cryptosystem, Hwang et al. [52] proposed a DLP based cryptosystem. Later, Lyuu et al. [53] developed an improved version

of above protocol. **Since, size of the keys of elliptic curve cryptosystem is smaller than other public key cryptosystem for a given level of security, we propose to develop protocols for encrypting large message using elliptic curve cryptosystem whose security is based on elliptic curve discrete logarithmic problem**.

3. Motivated by the use of Blind Signature in Information Security Services, such as digital cash, internet voting etc., several researchers developed protocol for Blind Signature. As mentioned in the literature survey 1.1.3, authors like Chaum [55], Sebastiaan et al. [72], Stadler et al. [74], Min-Shiang et al. [75] developed protocols whose security is based on integer factorization problem, authors like Fan et al. [80], Shao [58] developed protocols whose security is based on quadratic residue and authors like Camenisch et al. [84], Harn [86], Min-Shiang et al. [89], Lee et al. [90] developed protocols whose security is based on discrete logarithmic problem. **Considering the advantages of elliptic curve cryptosystem over above public key cryptosystem, we propose to develop Blind Signature schemes based on elliptic curve cryptosystem**.

4. Authentication is one of the services of Information Security System. As discussed in the Section 1.1.4, the remote user authentication problem was solved by Lamport [93], Mitchell et al. [96], Peyravian et al. [98], Lee et al. [99] using Hash function and Verifiable table. Later, Chang et al. [101], Yang et al. [102], Chan et al. [103] solved the same problem using Smartcard and Verifiable table. Yang et.al [102] and Hwang et al. [105] proposed schemes based on security of discrete logarithmic problem thereby eliminating the work of maintaining the verifiable table. **Motivated by this, we propose to develop protocols for remote user authentication using smart card which are based on security of elliptic curve discrete logarithmic problem and doesn't use Verifiable table**.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2- Key Agreement Protocol: In this chapter, a novel three party key agreement protocol has been proposed, which is based on SAKA. It begins with the explanation of existing protocols based on 2PKAP and 3PKAP. The proposed protocol withstands online and offline guessing attacks and does not use PKI.

Chapter 3- Large Message Encryption: Need of asymmetric protocol is presented in this chapter. A new protocol for encrypting large message based on ECDLP has been proposed.

Chapter 4- Digital and Blind Signature Schemes: Four schemes of Blind Signature based on ECDLP are proposed in this chapter. Computational time has been the major bottleneck in the existing literature. To overcome this shortcoming, the proposed schemes reduce the computational time by using ECDLP.

Chapter 5- Remote User Authentication Protocols: This chapter starts with brief description of existing protocol which uses DLP. Two protocols based on ECDLP which uses smart card have been proposed. These protocols uses ECDLP, hence they can be easily implemented in Smart card.

Chapter 6- Conclusion and Future Work: This chapter provides the concluding remarks with a stress on achievements and limitations of the proposed schemes. The scopes for further research are outlined at the end.

Chapter 2

Key Agreement Protocol

Chapter 2

Key Agreement Protocol

The primary objective of cryptography is to facilitate secure communication in an adversary environment. For instance, if two parties, A and B, want to safely communicate over an active network, they would definitely want to make sure that the data they correspond between themselves should remain private and authenticity of the data should be maintained. However, for this, there has to be a primitive cryptographic key agreement which contains both encryption and digital signature. With the help of such protocols carrying a common session key, two or more parties can exchange vital information over an adversely controlled and insecure network. These protected key agreement protocols act as the basic building block for assembling secure, intricate, higher-level protocols. Key establishment is generally segregated into key transport and key agreement section.

Secret keys offer message integrity and confidentiality where only trusted parties generally have copies of the secret key. However, in a global world of technological advancement, key distribution is a major problem from the security aspect. Basically, key establishment protocols need a set-up phase through which authentic and secret initial keying material is distributed. Most protocols are created with the objective of distinct keys on each protocol execution. In certain cases, the initial keying material pre-defines a fixed key which will result each time the protocol is implemented by a given pair or even a group of users. These static keys are insecure under known-key attacks.

There is a significant difference between key pre-distribution schemes and dynamic key establishment schemes. While key pre-distribution schemes handle pro-

tocols where initial keying material are the deciding factor to determine the established keys, dynamic key establishment schemes offer the convenience of choosing the established key by a fixed pair or group of users depending upon subsequent executions. As the session keys are dynamic in the later case, the protocols are immune to known-key threats. Many session key establishment protocols involve a trusted party, for both initial system setup and online actions involving real-time participation. Depending on the role played, this party is referred by a variety of names like trusted third party, authentication server, trusted server, key translation center (KTC), key distribution center (KDC), and certification authority.

To improve secure key establishment, the ideal way in a key establishment protocol is to determine the true identity of the sender and receiver which can be possible by gaining access to the resulting key and restrict any additional unauthorized parties from extracting the same key. For a secure key establishment, secrecy of the key and identification of accessing parties are required.

Motivation for Use of Session Keys

Key establishment protocols carrying the derived session keys are not secure as they can be stored in insecure ways, lost, stolen, forgotten or can even copied without authorization [108]. When a secret key that has been used many times, it won't give provide much authenticity about the secrecy of the key, which can really be a matter of concern. Ideally, the session key should be kept as a secret for a small time period, i.e., after a single session, all trace of the communication should be removed [109]. Motivation for ephemeral keys includes the following:

- To limit the existing ciphertext (under a fixed key) for cryptanalytic attack.
- To limit exposure in the event of compromise of session key with respect to both time as well as quantity of data.
- To evade long-term storage of a large number of distinct secret keys especially when one terminal communicates with a large number of systems, keys ought to be created when they are actually required.
- To create independency in applications or communication sessions so that

there would be a minimum requirement to maintain state information across sessions.

Challenges in Key Establishment Protocols

To simplify the threats, protocols subjected to an informal model for key establishment, inclusive of underlying assumptions is required. The definitions and models can be comprehensive, although attention is limited within two-party protocols. Communicating parties or entities in key establishment protocols having unique names are officially called principals. Along with legitimate parties, the existence of an unauthorized third party, which are otherwise known as impersonator, intruder, adversary, or opponent can also be hypothesized. Encryption algorithms and digital signature schemes are considered to be secure while examining the security of protocols. An adversary is not a cryptanalyst attacking the principal mechanisms directly, but rather one attempts to undermine the protocol objectives by defeating the manner in which such mechanisms are combined against attacking the protocol itself.

A passive attack involves an opponent trying to defeat a cryptographic technique by simply recording data like key establishment and subsequently analyzing it to verify the session key. But an active attack involves a challenger modifying or injecting messages. As protocol messages are transmitted over an open and unprotected network, it's possible for an adversary to insert, alter, delete, redirect, reorder, and reuse past or current messages and thus completely control the data therein. An active adversary can attack the whole network and even start up entirely new instances of players. It just needs to acquire session keys and corrupt players themselves to attack the network. In such circumstances, secure session key distribution is only possible when A and B have some information advantage over the invaders. To give emphasis to this, genuine parties are designed in such a way so as to receive messages exclusively via intervening adversaries through every communication path, which have the option of either relaying impassive messages to the intended recipients, or carrying out any of the above actions without any noticeable delay. Not only this, it is also capable of engaging gullible authorized parties in new protocol executions.

An adversary in a key establishment protocol may follow many strategies, including attempting to:

- Figure out a session key using information gained by eavesdropping.
- Take part covertly in a protocol initiated by one party with another, and influence it, e.g., by altering messages so as to be able to deduce the key.
- Commence off one or more protocol executions (possibly simultaneously), and combine (interleave) messages from one with another, so as to impersonate as some party or carry out one of the above attacks.
- Without being able to infer the session key itself, mislead a legitimate party regarding the identity of the party with which it shares a key. A protocol vulnerable to such an attack is not flexible.

The rest of this chapter is organized as follows. Sections 2.1 discusses 2-party key agreement protocol. 3-party key agreement protocol is explained in Section 2.2, a three party key agreement protocol based on one-way hash function has been proposed. in the Section 2.3. Finally, Section 2.4 summarizes results.

2.1 Two Party Key Agreement Protocol

In conventional key distribution protocols, an existing secure two-party authentication protocol (2PAP) [31] is considered as a stepping stone for building a series of simple and secure key distribution protocols. The protocols are devised to suit ideal security necessities, using the security properties of the underlying authentication protocol. This protocol is modular and simple. The following terminologies are used:

A, B, P, Q	Full principal names
S	Trusted Third Party
$E_k(X)$	Encryption of plaintext block X under key K
$MAC_K(X)$	Message Authentication Code
K_{ab}	A and B share Key K
N_{ab}	Nonce genrated by A and received by B
$A \rightarrow$B M	A sends message M to B

Initial Assumptions

Here it has been assumed that, there exists a secure 2PAP [31]. However, any secure nonce based 2PAP will be adequate for this purposes. Informally speaking, a 2PAP is regarded as protected if and only if it is computationally difficult for an intruder to masquerade as either party. The difficulty should be identical to the strength of the underlying cryptosystem or a strong one-way function. For instance, if DES is used with the 2PAP, the computational obscurity of defeating the protocol equals that of breaking DES by brute force, which is normally believed to require on the order of 2^{56} trials. 2PAP is as given below:

$$P \rightarrow Q \qquad P, N_{pq} \quad (1)$$

$$Q \rightarrow P \qquad AUTH_{K_{pq}}(N_{pq}, N_{qp}, Q), N_{qp} \quad (2)$$

$$P \rightarrow Q \qquad ACK_{K_{pq}}(Npq, Nqp, P) \quad (3)$$

In two-party key distribution protocol (2PKDP) [31], one of the parties initiates the protocol by requesting a new key. The other party acts in response by producing a new key and shipping it back to the requester. The protocol may include a verification flow whereby the initiator acknowledges the receipt of the new key. The 2PKDP is as specified below:

$$P \rightarrow Q \qquad P, N_{pq} \quad (1)$$

$$Q \rightarrow P \qquad AUTH_{K_{pq}}(N_{pq}, N_{qp}, Q) \oplus K_{new}, N_{qp} \quad (2)$$

This simple protocol consists of only two messages. This protocol has many similarities with 2PAP. Q generates K_{new} (session key) and sends to P. Given a secure 2PAP, it is computationally difficult for an intruder (not knowing K_{pq}) to obtain an $AUTH$ expression when at least one of the nonce N_{pq} or N_{qp} is selected by a genuine party. The $AUTH$ expression searched from the secure 2PAP. As both plaintext and key contain random and unpredictable values, this can be considered as strong encryption function. It resembles a truncated 2PAP except that the authentication expression in the second message is used as a one-time mask for the key being distributed.

Security Analysis

Key Disclosure— There are two sources of information which can be helpful for the attacker to break the protocol. In the first source, the attacker may record any number of legitimate executions of 2PKDP between P and Q. In such scenario, N_{pq} is always under control of P and N_{qp} is always under control of Q. On the other hand, he may try to masquerade as P by changing or composing a first message of the protocol and capturing Q's reply; where N_{pq} will be under the attacker's control, and N_{qp} will be selected by Q. In both cases, at least one of the nonces— N_{pq}, N_{qp}, is always under the control of a legitimate party, i.e., P or Q. Therefore, the ability to compute $AUTH_{K_{pq}}(N_{pq}, N_{qp}, Q)$ is equivalent to breaking 2PAP.

Key Modification is basically equivalent to key disclosure which is done to a value known by the attacker. If the attacker is able to change the key to a chosen value, then the corresponding $AUTH$ expression simultaneously becomes known. However, the attacker cannot know the key apriori; for that, he has to first know the $AUTH$ expression. That's why it's not possible to modify the key values.

Key Reuse involves the attacker feeding an old key to the initiating party. It might be noted here that it's not mandatory for the attacker to know the old key in order to try this attack. The simplest attack is to use pre-recorded replies from prior 2PKDP runs. Even though the attacker can convince P to accept any value G^X as an expression masking the key, the protocol maintains its strength since the key extracted from G^X remains secret.

Key Independence involves protocol runs to be distinct. Knowledge of a single session key cannot lead to the discovery of other session keys.

Key Integrity— This protocol does not provide key integrity. Key integrity is not necessarily considered a primary property of a secure key distribution protocol.

2.1.1 Diffie-Hellman Protocol

Two users can exchange a secret key over an insecure medium and without any prior secrets through this protocol. Today, this key exchange technique is employed in a number of commercial products. The algorithm itself is limited to the

exchange of keys. For its effectiveness, Diffie-Hellman algorithm depends on the difficulty of computing discrete logarithms. The Protocol is as follows:

Two parties A and B who wish to establish a key agree upon global public elements.Let be P a large prime number and g a generator over a finite (Galois) field. Let G be a cyclic group generated by g (primitive root of P) which is of order P. Then, every element of G can be expressed as g^n, $n \in [1, p-1]$, i.e, if g is a primitive root, powers of g generate all the integers from 1 to $p-1$. $g \pmod P$, $g^2 \pmod P$, $g^3 \pmod P$, ..., $g^{p-1} \pmod P$ are distinct and consist of the integers from 1 through $P-1$ in some permutation. We can find a unique exponent i such that $b \equiv g^i \pmod P$ where $0 \leq i \leq (P-1)$.

The exponent i is referred to as the discrete logarithm, or index of b for the base $g \bmod P$.

Step-1:

$A \rightarrow B \qquad Y_A = g^{X_A} \bmod P.$

User A selects a secret X_A (private) and $X_A \leq P-1$.

User A computes $Y_A = g^{X_A} (\bmod P)$.

A sends Y_A TO B.

Step-2:

$B \rightarrow A \qquad Y_B = g^{X_B} \bmod P.$

User B selects a secret X_B (private) and $X_B \leq P-1$.

User B computes $Y_B = g^{X_B} (\bmod P)$.

B sends Y_B TO A.

B computes session key as $K = (Y_A)^{X_B} (\bmod P)$

Step-3:

A computes session key as $K = (Y_B)^{X_A} (\bmod P)$

These two calculations of K produce identical results:

$$\begin{aligned} K &= Y_B^{X_A} (mod\ P) \\ &= (g^{X_B} \bmod P)^{XA} (\bmod P) \\ &= (g^{X_B})^{XA} (\bmod P) \\ &= (g^{X_A})^{XB} (\bmod P) \\ &= (g^{X_A} \bmod P)^{XB} (\bmod P) \\ &= Y_A^{X_B} (\bmod P) \end{aligned}$$

Attacks on Diffie-Hellman

Although useful, the Diffie-Hellman protocol is vulnerable to different attacks like [10, 110]:

- Man in the Middle Attack
- Degenerate Message Attack
- Simple Exponents Attack
- Simple Substitution Attack
- Identity Mis-binding Attack
- Subgroup Confinement Attack

2.1.2 Encrypted Key Exchange Protocol

This protocol allows two parties sharing a password to establish a secret key. The EKE [15] presents a novel and elegant method of key establishment.

Generic EKE

The generic version of EKE is illustrated below:

$$\begin{array}{lrl} A \rightarrow B & A, P(E_a) & (1) \\ B \rightarrow A & P(E_a(K)) & (2) \\ A \rightarrow B & K(C_a) & (3) \\ B \rightarrow A & K(C_a, C_b) & (4) \\ A \rightarrow B & K(C_b) & (5) \end{array}$$

Attacks on Generic EKE

The generic EKE [15] protocol is susceptible to Denning-Sacco Attack (DS) [111]. The attack proceeds as follows:

- The attacker manages to obtain one of the session keys used in one run of a key distribution protocol. Armed with that knowledge, the attacker is then able to impersonate one of the parties indefinitely often.

- The attacker somehow obtains one of the session keys distributed in one (recorded) run of EKE. Armed with that knowledge, the attacker mounts a dictionary attack on the password and, upon breaking the password, is able to impersonate one of the parties for an indefinite period.

2.1.3 SAKA Protocol

SAKA is a two-party key agreement protocol. This protocol is based on Diffie-Hellman key agreement. It is simple and cost effective as it has less computation cost and can be achieved in less number of steps. Here, Password based mechanism has been used for user authentication. The Protocol is as follows:

It is implicit that A and B (system principals) are to share the weak secret (password) pw in a secure way. They agree upon the generator g and its group Z_p^*. x and y are selected in Z_p^* for a uniform distribution, and $X = g^x(\text{mod}P)$ and $Y = g^y(\text{mod } P)$ are also in Z_p^* for a uniform distribution. The session key is made by $h(g^{xy} \text{ mod } P)$. The protocol run as follows.

1. $A \rightarrow B$ $\quad X \oplus pw$

A chooses a random number x, computes $X = g^x(\text{mod}P)$, encrypts it with pw and sends to B. After receiving message of step 1, B recovers X by using the password pw. Then, B chooses a random number y, computes $Y = g^y(\text{mod}P)$ and $Key_2 = X^y(\text{mod}P) = g^{xy}(\text{mod}P)$ like in Diffie-Hellman protocol. B encrypts Y with pw. B also computes one-way hash $h()$ using X, Key_2 as parameters. B sends $Y \oplus pw||h(Key_2, X)$ to A.

2. $B \rightarrow A$ $\quad Y \oplus pw||h(Key_2, X)$

After receiving message of step 2, A recovers Y by using the password pw and computes $Key_1 = Y^x(\text{mod}P) = g^{xy}(\text{mod}P)$. A also computes one-way hash $h()$ using X,Key_1 as parameters and verifies that $h(Key_1, X) = h(Key_2, X)$. If they match each other, A confirms that Key_2 is valid and both parties possess same key. It also suggests that X is not tampered in transit and Y is from valid source(B), i.e., it authenticates B. Then, A computes the response data $h(Key_1, Y)$ and sends it to B.

3. $A \rightarrow B$ $\quad h(Key_1, Y)$

B computes $h(Key_2, Y)$ and verifies $h(Key_1, Y) = h(Key_2, Y)$. If they match each

other, B confirms that Key_1 is valid and both parties possess same key. It is also suggested that Y is not tampered in transit and X is from valid source(A), i.e., it authenticates A.
Finally, A and B agree on the common session key $K = h(Key_1) = h(Key_2) = h(g^{xy} \bmod p)$.

Security Analysis

SAKA protocol is based on computational Diffie-Hellman problem, which states that computing $g^{xy}(modP)$ giving $g^x(modP)$ and $g^y(modP)$ is hard. SAKA [30] also satisfies completeness property, i.e., if each party's messages are faithfully relayed to one another, then the parties succeed in authentication and key agreement, at least with overwhelming probability. Without knowing the password, adversaries can not be accepted by the principals. When there are pieces of information in communications that can be used to verify the correctness of the guessed passwords, then only offline password guessing attack will be considered as succeeded. In the SAKA protocol, a passive attacker , all he receives from the protocol is as follows: $X \oplus pw$,$Y \oplus pw$,$h(Key_1, Y)$,$h(Key_2, X)$. He first guesses a password pw' and finds $g^{x^1} = X \oplus pw \oplus pw'$ and $g^{y^1} = X \oplus pw \oplus pw'$. If he wants to verify his guess, he has to find Key_1 or Key_2 which is impossible.

Since x and y are selected in the cyclic group for a uniform distribution, we can see that X and Y remain on the cyclic group under uniform distribution, and $X \oplus pw$ and $Y \oplus pw$ also remain on the cyclic group under uniform distribution. There is no way to find the relationship between the rejected password and the remaining password. On-line trial on password cannot partition out the possible set. The partitioning implies that the possible set decreases logarithmically. If an adversary tries to masquerade B and defraud A, she can know $g^x \oplus pw$ sent from A and $y, g^y, g^y \oplus pw'$ by herself where pw' is a guessed password. It is helpless since there is not any verifiable data. That means, he can not carry out the offline guessing attacks. To continue, he has to reply $h(Key_2, X)$ and it is not possible since he cannot find out Key_2.

SAKA offers entity authentication through shared passwords. This protocol takes minimum number of steps and random numbers and hence can be called as

optimal 2PKAP. Both the parties encrypt the data using shared secret. It is secure against dictionary attacks as there is no confirmable information present. It can be observed from results that one-way hash is not at all useful for cryptanalysis purpose as it is irreversible.

2.2 3-Party Key Agreement Protocol

Although the properties of the 2PKAP discussed so far appear reassuring, it's not a particularly useful application. Three-party key distribution offers much better common scenario [31]. Here two parties having no shared secret key enlist the assistance of a mutually trusted third party which performs the actual key distribution. This trusted third party is commonly referred as AS or KDC. Each of the two parties is assumed to share a long term key with the AS. The conditions for a secure 3PKAP are fundamentally similar to that of 2PKAP with the only exceptional requirement that a 3PKAP must be secure against a malicious insider. This insider is nothing but a legitimate party which can gather enough information to impersonate other parties or otherwise abuse the protocol (e.g., a malicious insider disclosing a key shared with another party).

A naive version of a secure 3PKAP has been illustrated here. It has been constructed by simply putting together two runs of 2PKAP.

$$A \rightarrow S \qquad A, B, N_{as} \quad (1)$$

$$S \rightarrow A \qquad AUTH_{K_{as}}(N_{as}, N_{sa}, B) \oplus K_{ab}, N_{sa} \quad (2)$$

$$B \rightarrow S \qquad B, A, N_{bs} \quad (3)$$

$$S \rightarrow B \qquad AUTH_{K_{bs}}(N_{bs}, N_{sb}, A) \oplus K_{ab}, N_{sb} \quad (4)$$

One notable aspect is that the key being distributed in messages 2 and 4 is one and the same-K_{ab}. The names of the parties involved are changed to highlight the difference with respect to previously discussed two-party protocols. A and B are the two principals and S is the mutually trusted AS. The only other aspect where the present protocol differs from 2PKDP is in the way principal names are used within $AUTH$ tokens. Whereas, before a name signified the originator of a token, it now refers to the third party in the protocol, e.g., the $AUTH$ token

sent from S to A includes B's name. Similarly, the $AUTH$ token sent from S to B includes A's name. This feature is essential to prevent masquerading attacks whereby a malicious party tampers with the principals' names in message of step 1 of the protocol. The protocol is secure with respect to outsider attacks, i.e., a non-participating party (except A, B or S) cannot subvert the protocol. This follows directly from the established security of 2PKDP.

Insider Attacks

The new danger introduced in this 3PKAP as a result of using the same key in messages of steps 2 and 4 are the so called insider attacks by either A or B. Both A and B, being privy to K_{ab} can discover each other's $AUTH$ expressions and try to use this new knowledge in some malicious fashion.

2.2.1 STW Protocol

This protocol was proposed by Steiner, Tsudik and Waidners [23]. Password-based mechanism allows people to choose their own passwords without any assistant device to generate or store and that's why it's an extensively used method for authentication. However, people are used to choose easy-to-remember passwords which prompts easy guessing to be a success. In a 3PKAP, all clients share their secrets with a trusted server only. This protocol is ideal for large communication environments. From the form of passwords stored in (B), there are two types of protocols, plaintext-equivalent protocols in which the clear form of the A's password is stored in B, and verifier-based protocols in which the verifier that is easily computed from the password, yet deriving the password from the verifier is computationally infeasible, is stored in B. The verifier-based protocol has the advantage that a compromised verifier does not reveal the password directly. However, a compromised verifier of a weak password can suffer from the guessing attack. In addition, the verifier-based protocol has more computational overheads than the plaintext-equivalent protocol. Hence, a secure plaintext-equivalent protocol is an ideal choice especially when we can't confine people to choose and remember strong passwords.

From session key creation prospective, such protocols can be classified into two

types— key transport protocols where the session key is created by one party and then securely transmitted to the other party, and key agreement protocols where both parties contribute information for creating the resultant session key. While key transport protocol is suitable for some special environments, key agreement protocols offers better security as none can fully control the session key. In key transport type of three-party protocols, instead of one of the two communication parties, the session key is created by the server S. This will result in the worry that a malicious server can get all transaction contents. In most applications, we only need the server to be an AS but not to be a monitor center (except some special needs, like the issue of national defense). In key agreement type of three-party protocols, the session key contributed from A, B and S needs more computational cost than it contributed from A and B. Moreover, it is still unknown to the server S.

Steiner, Tsudik, and Waidner proposed a 3PEKE protocol (STW 3PEKE) [23] and declared that it performed the following tasks:

- Secure distribution of session key K to A and B.
- Mutual authentication of A and B.
- (Indirect) authentication of S to A and S to B.

Every host shares a secret(password) with trusted third party denoted by P. They agree upon Diffie-Hellman parameters g(generator), p(large prime number). STW 3PEKE protocol is as given below:

1. $A \rightarrow B$ $\quad [R_A \oplus B]_{P_A}$

A chooses a random exponent N_A, keeps it secret and computes $R_A{=}g^{N_A}(\bmod p)$. Then, A encrypts $[R_A \oplus B]$ with his password P_A and sends the encrypted message as a request to B. After receiving A's request, B also chooses a random exponent N_B, keeps it secret, computes $R_B{=}g^{N_B}(\bmod p)$, then encrypts $[R_B \oplus A]$ with P_B. B forwards A's request with the encrypted message to S.

2. $B \rightarrow S$ $\quad$ A, $[R_A \oplus B]_{P_A}, [R_B \oplus A]_{P_B}$

S decrypts $[R_A \oplus B]_{P_A}, [R_B \oplus A]_{P_B}$ with P_A and P_B respectively and responses $R_A^{N_S}$, $R_B^{N_S}$ to B, in which N_S is a random exponent S chose.

3. $S \rightarrow B$ $\quad R_A^{N_S}$, $R_B^{N_S}$

B computes the session key $K = (R_A^{N_S})^{N_B} = g^{N_A.N_B.N_S} (\bmod p)$ and sends $R_B^{N_S}$ with a key confirmation message $[Step1]_K$ to A.

4. $B \rightarrow A$ $\quad R_B^{N_S}$,$[Step1]_K$

A computes the session key $K = (R_B^{N_S})^{N_A} = g^{N_A.N_B.N_S} (\bmod p)$. A decrypts $[Step1]_K$ with session key K and checks Step-1 equals $[R_A \oplus B]_{P_A}$ to confirm that B possesses the same session key K. A re-encrypts $[Step1]_K$ with the session key K and responses it to B for key confirmation.

5. $A \rightarrow B$ $\quad [[Step1]_K]_K$

B decrypts $[[Step1]_K]_K$ with the session key K and checks $[Step1]_K$ to confirm that A possesses the same session key K.

Undetectable Online Guessing Attacks

For instance, if an attacker attempts to use a guessed password in an online transaction, he would verify if the accuracy of his guess using responses of S. If his guess fails, he must start a new transaction with S using another guessed password. A failed guess can not be detected and logged by S, as S is not able to make track of an honest request from that of a malicious one. This can be demonstrated in the following two scenarios. In these two scenarios, the attacker B, who is valid but malicious, completes the protocol with S and no participation of A is required.

Scenario 1:

1. B: records $[R_A \oplus B]_{P_A}$

   The attacker B records $[R_A \oplus B]_{P_A}$ of an arbitrary run of the protocol. He guesses a password $\overline{P}_A$ and computes the value $\overline{R}_A$. He sets $R_B = \overline{R}_A$ and encrypts $\overline{R}_A \oplus A$ with his password P_B. Then, he sends S the message $A, [R_A \oplus B]_{P_A}, [\overline{R}_A \oplus A]_{P_B}$.

2. $B \rightarrow S$ A, $[R_A \oplus B]_{P_A}, [\overline{R}_A \oplus A]_{P_B}$
   S decrypts $[R_A \oplus B]_{P_A}, [\overline{R}_A \oplus A]_{P_B}$ with P_A and P_B respectively and responses $R_A^{N_S}$,$\overline{R}_A^{N_S}$ to B, in which N_s is a random exponent S chose.

3. $S \rightarrow B$ $R_A^{N_S}, \overline{R}_A^{N_S}$
   The attacker B compares the two values. If $R_A^{N_S} = \overline{R}_A^{N_S}$ and so he has guessed the correct password $\overline{P}_A = P_A$.

Scenario 2:

1. B: $[\overline{R}_A \oplus B]_{\overline{P}_A}$
   The attacker B guesses a password $\overline{P}_A$ generates on behalf of A a random exponent $\overline{N}_A$ and computes $\overline{R}_A = g^{\overline{N}_A} (\bmod\ P)$. Then, B encrypts $[\overline{R}_A \oplus B]$ with the guessed password $\overline{P}_A$. Additionally, B chooses a random exponent N_B computes $R_B = g^{N_B} (\bmod P)$ and encrypts $[R_B \oplus A]$ with his password P_B. Then, he sends S the message A,$[\overline{R}_A \oplus B]_{\overline{P}_A}$,$[R_B \oplus A]_{P_B}$.

2. $B \rightarrow S$ A, $[\overline{R}_A \oplus B]_{\overline{P}_A}, [R_B \oplus A]_{P_B}$
   S decrypts $[\overline{R}_A \oplus B]_{\overline{P}_A}, [R_B \oplus A]_{P_B}$ with P_A and P_B respectively and responses $R_A^{N_S}$, $R_B^{N_S}$ to B. in which N_S is a random exponent S chose.

3. $S \rightarrow B$ $R_A^{N_S}, R_B^{N_S}$
   The attacker B computes the two values $(R_A^{N_S})^{N_B}$ and $(R_B^{N_S})^{\overline{N}_A}$. If they are equal, it follows that he has guessed the correct password $\overline{P}_A = P_A$.

2.2.2 LSH 3PEKE Protocol

This protocol was proposed by Chun-Li Lin et al. [25]. This protocol is secure against both offline as well as undetectable online guessing attacks [20]. It also justifies the security properties of perfect forward secrecy. The most important prerequisite to prevent undetectable online guessing attacks is to provide authentication of A and B to S (server). In step 2 of the STW 3PEKE, the message $[R_A \oplus B]_{P_A}, [R_B \oplus A]_{P_B}$ doesn't contain any verifiable information for S to authenticate A and B even though S uses correct passwords to decrypt that message. On the contrary, if there is any verifiable information for S combined with password P_A and P_B, it will result in offline guessing attacks. One solution for this problem

could be through the help of server S's public key. Assuming that S's public key is a cryptographic parameter which is secure against guessing and exhaustive attacks, and is well-known for all parties. Thus, any verifiable information with a confounder encrypted with S's public key is able to resist offline guessing attacks. Based on such idea, a new 3PEKE protocol was proposed which is secure against both offline guessing attacks and undetectable online guessing attacks [20].

The LSH 3PEKE is described below:

1. $A \rightarrow B$ $\quad A, \{ra, R_A, P_A\}_{K_s}$

   A chooses a confounder ra and a random exponent N_A, keeps them secret and computes $R_A = g^{N_A}(\mathrm{mod}P)$. Then, A encrypts ra, R_A, P_A with server's public key K_s and sends the encrypted message as a request to B. After receiving A's request, B also chooses a confounder rb and a random exponent N_B, keeps them secret, computes $R_B = g^{N_B}(\mathrm{mod}P)$ then encrypts rb, R_B, P_B with server's public key K_s. B forwards A's request with the encrypted message to S.

2. $B \rightarrow S$ $\quad A, \{ra, R_A, P_A\}_{K_s}, \{rb, R_B, P_B\}_{K_s}$

   S decrypts $\{ra, R_A, P_A\}_{K_s}, \{rb, R_B, P_B\}_{K_s}$ with his private key and authenticates A and B by verifying their passwords P_A and P_B respectively. Then, S encrypts B, R_B with ra, encrypts A, R_A with rb and sends them to B. Notice that the values ra and rb also act as one-time keys.

3. $S \rightarrow B$ $\quad [B, R_B]_{ra}, [A, R_A]_{rb}$

   B decrypts $[A, R_A]_{rb}$ with rb and authenticates S by verifying the integrity of ID A. B computes the session key K=$R_A^{N_B}(\mathrm{mod}P)$ and sends $[B, R_B]_{ra}$ with a key-confirmation message $[f(Step1), C_B]_K$ to A.

4. $B \rightarrow A$ $\quad [B, R_B]_{ra}, [f(Step1), C_B]_K$

   A decrypts $[B, R_B]_{ra}$ with ra and authenticates S by verifying the integrity of ID B. A computes the session key K=$R_B^{N_A}(\ \mathrm{mod}\ P)$ decrypts $[f(Step1), C_B]_K$ with the session key K and checks whether $f(Step1) = f(A, \{ra, R_A, P_A\}_{K_s})$ to confirm that B possessed the same session key K. Then, A responses C_B to B for key confirmation.

5. $A \rightarrow B \qquad C_B$

   B checks C_B to confirm that A possessed the same session key K.

Public key infrastructure has been used for this protocol through the RSA algorithm. In the first two steps of protocol all data are encrypted using server's public key using RSA algorithm. This protocol provides key confirmation also in last two steps. All data are not vulnerable to dictionary attacks as they have been encrypted using server's public key.

2.3 Proposed 3-Party Key Agreement Protocol

Key agreement protocols proposed on password based mechanism are vulnerable to dictionary attacks. Storing plaintext version of password on server is always not a viable option always. To withstand all online and offline guessing attacks, a new 3PKAP is proposed, where PKI usage is not required. In this protocol, a trusted third party KDC has been used which intermediates in key distribution. Here, one-way hash of the password are stored instead of storing plaintext version of password. For session key, every host and server agree upon family of commutative hash functions, using which host can authenticate itself to server. During this protocol run, host establishes one time key with server using which server also authenticates the host. Moreover, any public key infrastructure has not been used which needs large computational power. Since this is a 3PKAP, every host need not share secret information with every other host.

Before we proceed we define one-way hash function:
A one-way function is a function f such that for each x in the domain of f, it is easy to compute $f(x)$, but for essentially all y in the range of f, it is computationally infeasible to find any x such that $y = f(x)$.

The protocol is as described below:

1. $A \rightarrow S \qquad A, B, H(P_A)[R_A]$

   A chooses a random number ra and generates $R_A = g^{ra}(\mathrm{mod} PP)$ like in Diffie-Hellman protocol, where g is a generator of cyclic group and P is a large prime. It also generates one-way hash of its password $H(P_A)$. A

encrypts R_A with $H(P_A)$ and sends it to server S along with IDs of participating entities. Server stores one-way hash of password of every host (assumed to be pre distributed), using which it decrypts the above packet to get $R_A = g^{ra}(mod\ P)$.

2. $S \rightarrow A$ $\quad H(P_A)(g^{rs_1} \bmod P), H(P_B)(g^{rs_2} \bmod P)$
   Server chooses random numbers rs_1 and rs_2. S generates $g^{rs_1}(\text{mod}P)$ and $g^{rs_2}(\bmod P)$ respectively. Using these quantities server establishes ephemeral keys with A and B respectively. Using this ephemeral keys S authenticates itself to A and B. S computes these ephemeral keys as specified below:
   $K_{AS} = (g^{ra})^{rs_1} \bmod P$
   $K_{BS} = (g^{rb})^{rs_2} \bmod P$
   Note that K_{BS} can be computed only after fourth step of the protocol.
   $g^{rs_1}(\bmod P)$ and $g^{rs_2}(\bmod P)$ are encrypted with $H(P_A)$ and $H(P_B)$ respectively and dispatched to A. A decrypts this packet with $H(P_A)$.

3. $A \rightarrow B$ $\quad F_A(P_A, K_{AS})\ , H(P_B)(g^{rs_2} \bmod P)$ A decrypts this packet with $H(P_A)$ to get $g^{rs_1}(\text{mod}P)$. A establishes ephemeral key with S as $K_{AS} = (g^{rs_1})^{ra} \bmod P$. A calculates its predicate function $F_A(P_A, K_{AS})$ using which it authenticates itself to server. Since only A knows P_A only it can compute this predicate function. This is a commutative one-way hash function which is explained in Section 2.3. This value along with $H(P_B)(g^{rs_2} \bmod P)$ is forwarded to B. B decrypts with $H(P_B)$ to get $(g^{rs_2} mod\ P)$.

4. $B \rightarrow S$ $\quad F_A(P_A, K_{AS}), F_B(P_B, K_{BS}), H(P_B)[R_B]$
   B chooses a random number rb and generates $R_B = g^{rb}(\text{mod}P)$. It also generates one-way hash of its password $H(P_B)$. B computes ephemeral key for authenticating server as $K_{BS} = (g^{rs_2})^{rb} \bmod P$. B calculates its predicate function $F_B(P_B, K_{BS})$ using which it authenticates itself to server. This predicate function is a commutative one-way hash function. Password of B and ephemeral session key K_{BS} are seeds for this function. Since only B knows P_B, only it can compute this predicate function. After receiving this

packet, server decrypts with $H(P_B)$ to get R_B. Server computes ephemeral one time session key $K_{BS} = (g^{rb})^{rs_2} \bmod P$. Using this K_{AS},K_{BS}, server authenticates itself to hosts. K_{AS},K_{BS} changes with every protocol run. Server recomputes predicate functions of A and B, i.e., $F_A()$,$F_B()$ and authenticates A and B respectively.

Server need not know P_A to compute this predicate function. Since this is a commutative hash function and $H(P_A)$ is pre distributed, using $K_{AS}, H(P_A)$ server can calculate the predicate function $F_A()$ and authenticates A. Similarly, using $K_{BS}, H(P_B)$ server can calculate the predicate function $F_B()$ and authenticates B.

5. $S \rightarrow B$ $\quad f_{K_{AS}}(R_B), f_{K_{BS}}(R_A), H_{K_{AS}}(R_A, R_B), H_{K_{BS}}(R_A, R_B)$

   S encrypts R_B and R_A with K_{AS} and K_{BS} respectively. This defeats identity mis-binding attacks. $f()$is a cryptographic transformation function. S computes one-way hash function$H_{K_{AS}}(R_A, R_B)$ using K_{AS}(one time key shared between A and server)using this host-A authenticates the server. Similarly, S computes one-way hash function $H_{K_{BS}}(R_A, R_B)$ using K_{BS} (one time key shared between B and server). Using this, host-B authenticates the server.

   $f_{K_{AS}}(R_B), f_{K_{BS}}(R_A), H_{K_{AS}}(R_A, R_B), H_{K_{BS}}(R_A, R_B)$ are sent to B. After receiving this, B decrypts $f_{K_{BS}}(R_A)$ with K_{BS} and gets R_A. Since K_{BS} is shared between server and B, it ensures B that R_A value is from authentic source. B recomputes one-way hash $H_{K_{BS}}(R_A, R_B)$ using K_{BS} as key and authenticates server. B computes session key with A as $K_{AB} = (R_A)^{rb}(\bmod P)$ like in Diffie-Hellman protocol.

6. $B \rightarrow A$ $\quad f_{K_{AS}}(R_B), H_{K_{AS}}(R_A, R_B), H_{K_{AB}}(N_{AB}), N_{AB}$

   B forwards $f_{K_{AS}}(R_B), H_{K_{AS}}$ to A. A decrypts $f_{K_{AS}}(R_B)$ using K_{AS} to get R_B. Since K_{AS} is shared between server and A, it ensures A that R_B value is from authentic source. A computes session key with B as $K_{AB} = (R_B)^{ra}(\text{mod}P)$ like in Diffie-Hellman protocol. B also computes a one-way hash $H_{K_{AB}}(N_{AB})$ using K_{AB} and N_{AB} as seeds. Where N_{AB} is a random number. This one-way hash is used for key confirmation (assures

that both parties possess same session key). Since N_{AB} is transmitted in plain there is no need of decryption. One-way hash suffices decryption.

7. $A \rightarrow B \qquad H_{K_{AB}}(H_{K_{AB}}(N_{AB}))$

   Using K_{AB} and N_{AB} A recomputes one-way hash $H_{K_{AB}}(N_{AB})$ and verifies that B possesses same key, K_{AB} as A. Using K_{AB} A once again calculates one-way hash $H_{K_{AB}}(H_{K_{AB}}(N_{AB}))$ and sends to B. After receiving this B recomputes this one-way hash using K_{AB} and verifies that A possesses same session key(K_{AB}) as B.

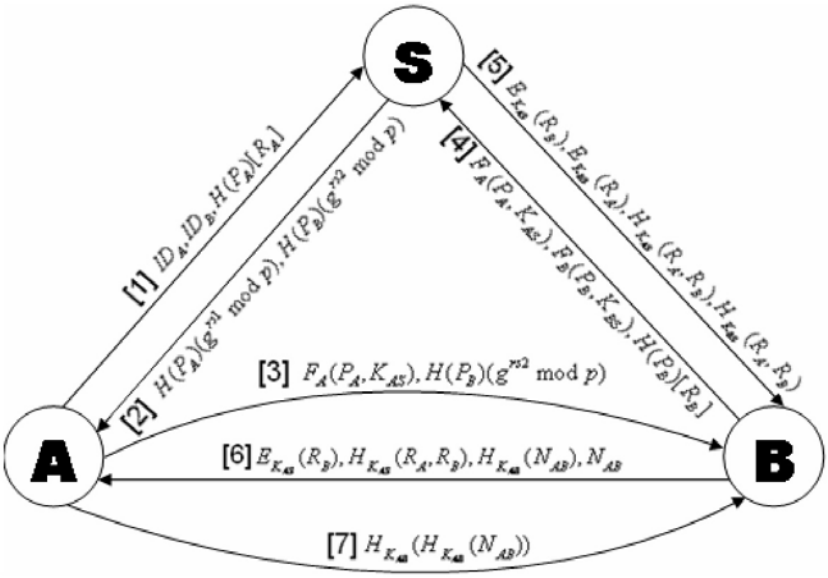


Figure 2.1: Proposed 3-Party Key Agreement Protocol

Commutative One-Way Hash Functions

Both host and server agree upon family of commutative hash functions $\{H_0, H_1, ..., H_n\}$. Let $H(P)$ be defined as $H_0(P)$, a member of a family of Commutative one-way hash functions. Host A calculates one way hash of its password as $H_0(P_A) = P_A^{h_0} (\text{mod } P)$, where, h_0 is a random number(which it keeps as secret). We assume that one way hash of password $H_0(P)$ of every host is distributed to server. Since one way hash is irreversible nobody can compute P from $H_0(P)$. Host A calculates its predicate function $F_A(\ )$ as :
$H_0(H_{K_{AS}}(P_A)) = (P_A^{K_{AS}})^{h_0} (\text{mod } P)$.

Server Knows only one way hash of password P_A, i.e., $H_0(P_A)$ using which it calculates predicate function of A as $H_{K_{AS}}(H_0(P_A))$.
$H_{K_{AS}}(H_0(P_A)) = (P_A^{h_0})^{K_{AS}} (\text{mod } P)$
Here K_{AS} is one time (ephemeral) key established between server and A. Since server knows K_{AS} and $H_0(P_A)$ it can compute this predicate function and authenticate A. Similarly it computes predicate function for B and authenticates B. Since these are commutative hash functions $H_{K_{AS}}(H_0(P_A)) = H_0(H_{K_{AS}}(P_A))$, i.e., $(P_A^{K_{AS}})^{h_0} (\text{mod } P) = (P_A^{h_0})^{K_{AS}} (\text{mod } P)$.

Security Analysis

Proposed protocol extends Two party version of SAKA to 3-party protocol. Every host need not share a secret with every other host. As all the transactions in the protocol are done using one-way hash of the password, host is not required to store plaintext version of its password at server, which consequently defeats dictionary attacks. Since this is a one-way hash function, there is no way to recover P from $H(P)$. Every time, host and server establish a one-time (ephemeral) key using which host authenticates server. Unlike traditional 3PKAP, we need not use long term or master keys which often lead to malicious insider attacks. In malicious insider attacks, one of the contributing parties turns aggressive and misuses the information it has acquired in previous protocol runs and breaks the system. Through the predicate functions, a server authenticates the hosts. For calculating predicate functions, server need not know password of hosts. It can be calculated by using one-way hash of the password of the hosts.

Under some special cases, even though $H(P)$ is compromised, nobody can mimic the host to server as only legitimate hosts can calculate predicate functions. Nobody can mimic the server to the host even if $H(P)$ is compromised; it's because except the host, nobody knows the password which can be a seed for predicate function. It is equivalent to breaking Diffie-Hellman problem. S encrypts R_B and R_A with K_{AS} and K_{BS} (one time ephemeral keys) respectively. This defeats identity mis-binding or masquerading attacks. Here the server acts just like AS and not as monitoring server. This prevents malicious server from knowing session key and subsequently knowing all transactions. This protocol also ensures key

integrity, key non disclosure, and key confirmation. Proposed protocol also ensures perfect forward key secrecy. Even if one of the session keys are compromised, it will not lead to disclose of future keys. This protocol sustains online and offline guessing attacks as there is no verifiable information present.

2.4 Results and Discussion

The proposed protocol is simulated with Java using an Intel Core 2 Duo Processor. It can be observed that the password of host data is stored in encrypted form by using the one-way hash. Predicate function is calculated using commutative hash functions using password as seed at client side and one-way hash of the password at server. Using session key and a random number as seeds the key confirmation is provided. By including the random numbers, the key space in case of dictionary attacks will be widened. Encryption of R_B and R_A using one-time keys provides user authentication and stops malicious insider attacks. It is compared with well-known 3PKAP which is given in Table 2.1.

Table 2.1: Comparative statement among 3-party key agreement protocols

Protocols	Strengths/Weaknesses
STW	Vulnerable to offline and online guessing attacks.
LHS	Uses expensive public key infrastructure.
Proposed	Defeats offline and online guessing attacks, dictionary attack, misbinding attack, etc. and does not use public key infrastructure. ***It uses one-time key, commutative hash functions. Server encrypts*** R_A ***and*** R_B ***with*** K_{BS} ***and*** K_{AS} ***respectively.***

In proposed protocol, public key infrastructure is not used. Since this is a 3-party protocol every host need not share a secret with other host. The protocol provides host authentication and server authentication as a result man-in-the middle attacks are averted. It also spoils online and off-line guessing attacks as it uses one time keys, commutative hash functions for authentication. Proposed protocol ensures perfect forward key secrecy, key integrity. The protocol also sustains malicious insider attacks as it uses one time keys for authentication. Server acts just like AS and not like a monitoring server.

Chapter 3

Large Message Encryption Protocol

Chapter 3

Large Message Encryption Protocol

Cryptography (from Greek words *kryptos*, meaning hidden, and *graphia*, meaning writing) is the science of encrypting and decrypting communications to make them unitenlligible for all but the intended recipient. Hence, cryptography plays an important role for achieving information security in communications, computer systems, electronic commerce, and, more generally, in the emerging information society.

Throughout the history of cryptography, hundreds of cryptosystems have been developed. The earliest ones, as well as many later ones, relied on the complete secrecy in transferring keys between the sender and recipient. These kinds of systems, called secret key cryptosystem, have just a single key which is used for both encryption and decryption; therefore, these systems are more frequently known as symmetric cryptosystems which is shown in Figure 3.1 [2].

The advantages and disadvantages of symmetric cryptosystem are given below:

Advantages

- A symmetric cryptosystem is faster.
- In Symmetric Cryptosystems, encrypted data can be transferred on the link even if there is a possibility that the data will be intercepted. Since there is no key transmitted with the data, the chances of data being decrypted are null.

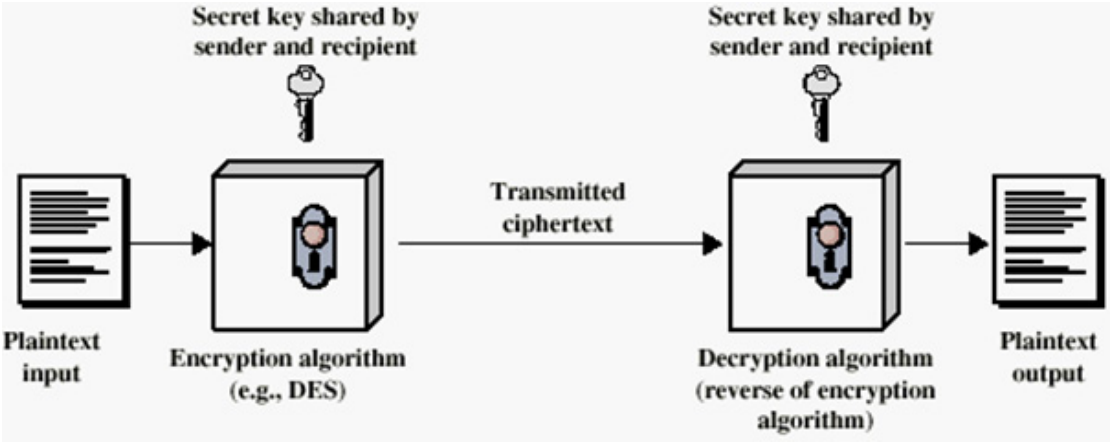


Figure 3.1: Symmetric Key Cryptosystem [2]

- A symmetric cryptosystem uses password authentication to prove the receiver's identity.
- A system only which possesses the secret key can decrypt a message.

Disadvantages

- Symmetric cryptosystems have a problem of key transportation. The secret key is to be transmitted to the receiving system before the actual message is to be transmitted. Every means of electronic communication is insecure as it is impossible to guarantee that no one will be able to tap communication channels. So the only secure way of exchanging keys would be exchanging them personally.
- Cannot provide digital signatures that cannot be repudiated.

The problems of key distribution are solved by public key cryptography, the concept of which was introduced by Whitfield Diffie and Martin Hellman in 1976. The primary benefit of public key cryptography is that it allows people who have no preexisting security arrangement to exchange messages securely. The need for sender and receiver to share secret keys via some secure channel is eliminated; all communications involve only public keys, and no private key is ever transmitted or shared. In contrast to a symmetric cryptosystem, an asymmetric cryptosystem is associated with a pair of keys; therefore, not everything related to the cryptosystem needs to be kept secret. This kind of a cryptosystem is also called public-key

cryptosystem. A public key cryptosystem consists of a public key, which is used for encryption, and a private key, used for decryption. Therefore, anybody can encrypt plaintext since the encryption key is available to everyone, but only the holders of the private key corresponding to the public key used for encryption can decrypt the ciphertext. Public key cryptosystems are based on mathematical problem, e.g., IFP, DLP and ECDLP etc. Some examples of public-key cryptosystems are Elgamal, Diffie-Hellman and Digital Signature Algorithm (DSA) based on DLP and RSA based on IFP which is shown in Figure 3.2 [2].

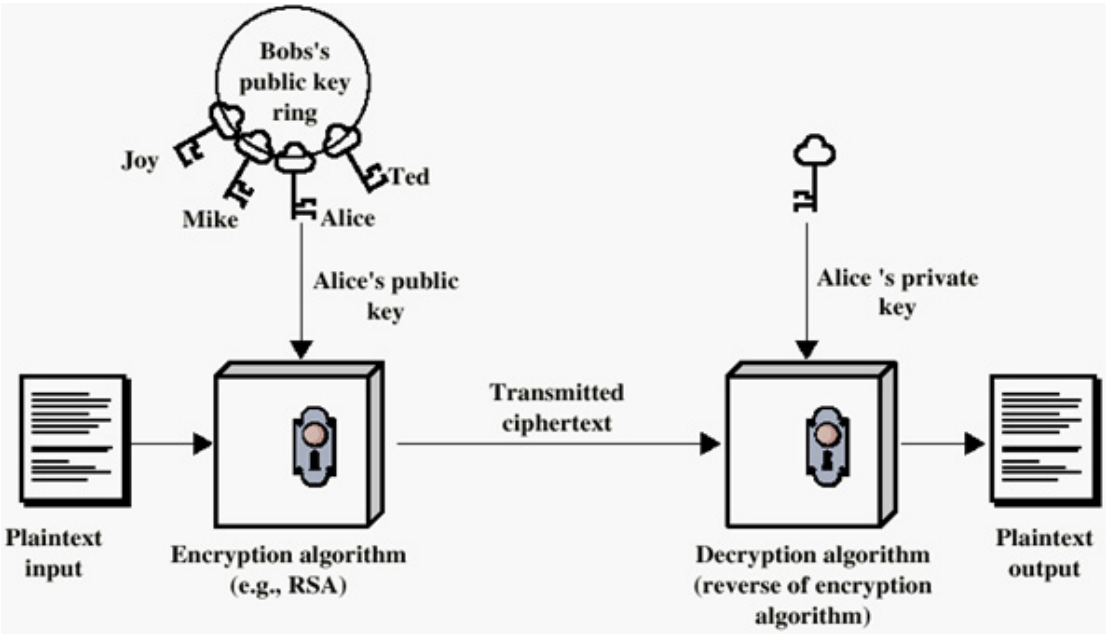


Figure 3.2: Asymmetric Key Cryptosystem [2]

The advantages and disadvantages of asymmetric cryptosystem are given below:

Advantages

- In asymmetric or public key, cryptography there is no need for exchanging keys, thus eliminating the key distribution problem.
- The primary advantage of public-key cryptography is increased security—the private keys do not ever need to be transmitted or revealed to anyone.
- Can provide digital signatures that can be repudiated.

Disadvantage

- A disadvantage of using public-key cryptography for encryption is speed—there are popular secret-key encryption methods which are significantly faster than any currently available public-key encryption method.

By looking at above discussion, to take the advantage of both categories of cryptosystem, a symmetric cryptosystem is used in addition to an asymmetric cryptosystem, if there is a need to encrypt a large message. So, traditionally, message is encrypted using the symmetric cryptosystem and the symmetric key is encrypted using the asymmetric cryptosystem which is shown in Figure 3.3.

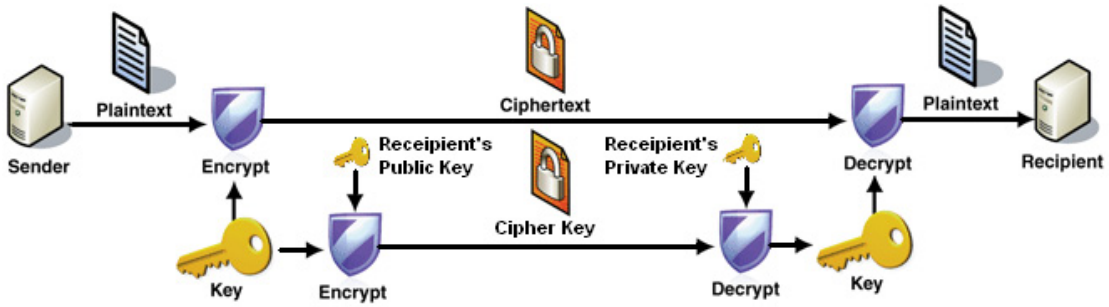


Figure 3.3: Asymmetric Key Cryptosystem

To eliminate the requirement for an additional cryptosystem, a novel asymmetric cryptosystem based on ECDLP has been proposed. With less bit size key, elliptic curve cryptosystem gives more security and is computationally faster than the other asymmetric cryptosystems and hence can be used for encrypting large message based on ECDLP [112–114].

The rest of this chapter is as follows. In the Section 3.1, the basic concept of elliptic curve cryptography (ECC) is explained. In Section 3.1.3, discussion on elliptic curve (EC) cryptosystem based on ElGamal scheme has been illustrated. The proposed scheme and its security analysis are discussed in section 3.3. Finally, Section 3.4 presents the results and discussions.

3.1 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a public key cryptography [115]. The use of ECC was initially suggested by Neal Koblitz [113] and Victor S. Miller [112] and

there after many researchers have suggested different application of Elliptic Curve Cryptosystems. EC cryptosystems over finite fields have some definite advantages. One advantage is the much smaller key size as compared to other cryptosystems like RSA or Diffie-Hellman, since (a) only exponential-time attack is known so far if the curve is carefully chosen [114], and (b) elliptic curve discrete logarithms might be still intractable even if factoring and multiplicative group discrete logarithms are broken. Further, ECC is also more computationally efficient than the first-generation public key systems such as RSA or Diffie-Hellman [116].

The mathematical operations of ECC is defined over the elliptic curve as,

$$y^2 = x^3 + ax + b$$

where, $4a^3 + 27b^2 \neq 0$. Each value of a and b gives a different elliptic curve. All the points (x, y) which satisfy the above equation and a point at infinity lies on the elliptic curve. The public key is a point in the curve and the private key is a random number. The public key is obtained by multiplying the private key with the generator point G in the curve. The generator point G, the curve parameters a and b, together with few more constants constitutes the domain parameter of ECC. The elliptic curve discrete logarithm problem is defined as follows [2].

***Definition*:** Let E be an elliptic curve over a finite field F_p and let $G \in E(F_p)$ be a point of order n. Given $Q \in E(F_p)$, the elliptic curve discrete logarithm problem is to find the integer $d \in [0, n-1]$, such that $Q = dG$.

In point multiplication, a point P on the elliptic curve is multiplied with a scalar k using elliptic curve equation to obtain another point Q on the same elliptic curve, i.e.,

$$Q = kP$$

Point multiplication is achieved by two basic elliptic curve operations— point addition and point doubling. These are explained below.

1. Point Addition

Point addition is the addition of two points J and K on an elliptic curve to obtain another point L on the same elliptic curve.

Geometrical Explanation

Consider two points J and K on an elliptic curve as shown in figure 3.4(a). If $K \neq -J$ then, a line drawn through the points J and K will intersect the elliptic curve at exactly one more point $-L$. The reflection of the point $-L$ with respect to X-axis gives the point L, which is the result of addition of points J and K, i.e.,

$$L = J + K$$

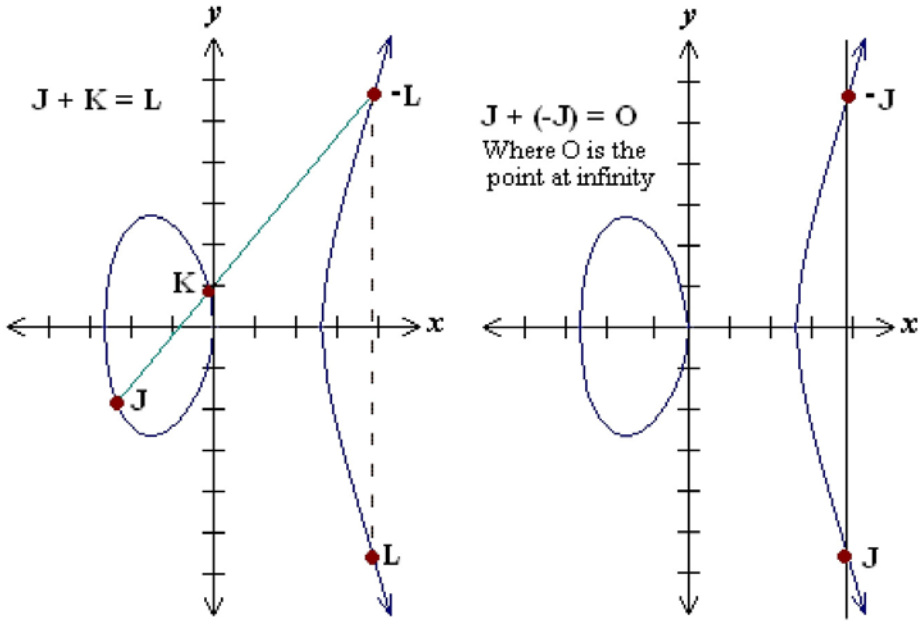


Figure 3.4: Point Addition (a) $J + K = L$, (b) $J + (-J) = O$

If $K = -J$, the line through this point intersect at a point at infinity O. Hence, $J + (-J) = O$. This is shown in figure 3.4(b). O is the additive identity of the elliptic curve group. A negative of a point is the reflection of that point with respect to X-axis.

Analytical Explanation

Consider two distinct points J and K such that $J = (x_J, y_J)$ and $K = (x_K, y_K)$. Let $L = J + K$ where, $L = (x_L, y_L)$. Then,

$$\begin{aligned} x_L &= s^2 - x_J - x_K \\ y_L &= -y_J + s(x_J - x_L) \\ s &= (y_J - y_K)/(x_J - x_K) \end{aligned} \tag{3.1}$$

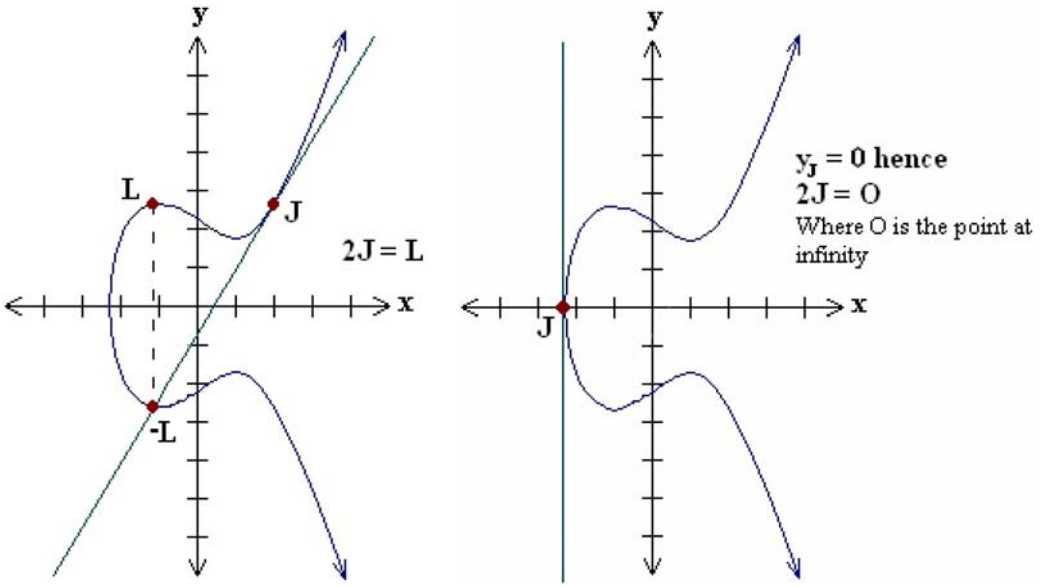
where, s is the slope of the line through J and K. If $K = -J$, i.e., $K = (x_J, -y_J)$ then, $J + K = O$ where, O is the point at infinity. If $K = J$ then, $J + K = 2J$; point doubling equations are used. Also $J + K = K + J$.

2. Point Doubling

Point doubling is the addition of a point J on the elliptic curve to itself to obtain another point L on the same elliptic curve.

Geometrical Explanation

To double a point J to get L, i.e., to find $L = 2J$, consider a point J on an elliptic curve as shown in figure 3.5(a). If y coordinate of the point J is not zero then, the tangent line at J will intersect the elliptic curve at exactly one more point $-L$. The reflection of the point $-L$ with respect to x-axis gives the point L, which is the result of doubling the point J. Thus, $L = 2J$.



(a) (b)

Figure 3.5: Point Doubling (a) $2J = L$, (b) $2J = O$ (when $y_J = 0$)

If y coordinate of the point J is zero then the tangent at this point intersects at a point at infinity O. Hence, $2J = O$ when $y_J = 0$. This is shown in figure 3.5(b).

Analytical Explanation

Consider a point J such that $J = (x_J, y_J)$ where, $y_J \neq 0$. Let $L = 2J$ where $L = (x_L, y_L)$. Then,

$$\begin{aligned} x_L &= s^2 - 2x_J \\ y_L &= -y_J + s(x_J - x_L) \\ s &= (3(x_J)^2 + a)/(2y_J) \end{aligned} \tag{3.2}$$

where, s is the tangent at point J and a is one of the parameters chosen with the elliptic curve. If $y_J = 0$ then, $2J = O$ where, O is the point at infinity.

3.1.1 Elliptic Curve over Finite Field

The elliptic curve operations defined above are on real numbers. Operations over the real numbers are slow and inaccurate due to round-off error. Cryptographic operations need to be faster and accurate. To make operations on elliptic curve accurate and more efficient, ECC is defined over two finite fields— prime field F_p and binary field F_2^m. The field is chosen with finitely large number of points suited for cryptographic operations.

EC over Prime Field F_p

The equation of the elliptic curve on a prime field F_p is

$$y^2 \bmod p = (x^3 + ax + b) \bmod p$$

where, $(4a^3 + 27b^2) \bmod p \neq 0$. Here, the elements of the finite field are integers between 0 and $p - 1$. All the operations such as addition, subtraction, division, and multiplication involves integers between 0 and $p - 1$. The prime number p is chosen such that there is finitely large number of points on the elliptic curve to make the cryptosystem secure. Standards for Efficient Cryptography (SEC) specifies curves with p ranging between 112-521 bits [117, 118]. The algebraic rules for point addition and point doubling can be adapted for elliptic curves over F_p.

Point Addition

Consider two distinct points J and K such that $J = (x_J, y_J)$ and $K = (x_K, y_K)$. Let $L = J + K$ where, $L = (x_L, y_L)$ then,

$$\begin{aligned} x_L &= (s^2 - x_J - x_K) \bmod p \\ y_L &= (-y_J + s(x_J - x_L)) \bmod p \\ s &= (y_J - y_K)/(x_J - x_K) \bmod p \end{aligned} \tag{3.3}$$

where, s is the slope of the line through $J$and K. If $K = -J$, i.e., $K = (x_J, -y_J) \bmod p$ then $J + K = O$ where, O is the point at infinity. If $K = J$ then, $J + K = 2J$; point doubling equations are used. Also, $J + K = K + J$.

Point Subtraction

Consider two distinct points J and K such that $J = (x_J, y_J)$ and $K = (x_K, y_K)$. Then, $J - K = J + (-K)$ where, $-K = (x_k, -y_k) \bmod p$.

Point Doubling

Consider a point J such that $J = (x_J, y_J)$ where, $y_J \neq 0$. Let $L = 2J$ where, $L = (x_L, y_L)$. Then,

$$\begin{aligned} x_L &= (s^2 - 2x_J) \bmod p \\ y_L &= (-y_J + s(x_J - x_L)) \bmod p \\ s &= (3x_J^2 + a)/(2y_J) \bmod p \end{aligned} \tag{3.4}$$

where, s is the tangent at point J and a is one of the parameters chosen with the elliptic curve. If $y_J = 0$ then, $2J = O$ where, O is the point at infinity.

EC over Binary Field F_2^m

The equation of the elliptic curve on a binary field F_2^m is $y^2 + xy = x^3 + ax^2 + b$, where, $b \neq 0$. Here, the elements of the finite field are integers of length at most m bits. These numbers can be considered as a binary polynomial of degree $m-1$. In binary polynomial, the coefficients can only be 0 or 1. All the operation such as addition, subtraction, division, and multiplication involves polynomials of degree $m - 1$ or lesser. m is chosen such that there is finitely large number of points on the elliptic curve to make the cryptosystem secure. SEC specifies curves with

m ranging between 113-571 bits [118]. The algebraic rules for point addition and point doubling can be adapted for elliptic curves over F_2^m.

Point Addition

Consider two distinct points J and K such that $J = (x_J, y_J)$ and $K = (x_K, y_K)$. Let $L = J + K$ where, $L = (x_L, y_L)$. Then,

$$\begin{aligned} x_L &= s^2 + s + x_J + x_K + a \\ y_L &= s(x_J + x_L) + x_L + y_J \\ s &= (y_J + y_K)/(x_J + x_K) \end{aligned} \tag{3.5}$$

where, s is the slope of the line through J and K. If $K = -J$, i.e., $K = (x_J, x_J+y_J)$ then, $J + K = O$ where, O is the point at infinity. If $K = J$ then, $J + K = 2J$; point doubling equations are used. Also, $J + K = K + J$.

Point Subtraction

Consider two distinct points J and K such that $J = (x_J, y_J)$ and $K = (x_K, y_K)$. Then, $J - K = J + (-K)$ where, $-K = (x_k, x_k + y_k)$.

Point Doubling

Consider a point J such that $J = (x_J, y_J)$, where $x_J \neq 0$. Let $L = 2J$ where $L = (x_L, y_L)$. Then,

$$\begin{aligned} x_L &= s^2 + s + a \\ y_L &= (x_J)^2 + (s + 1) * x_L \\ s &= x_J + y_J/x_J \end{aligned} \tag{3.6}$$

where, s is the tangent at point J and a is one of the parameters chosen with the elliptic curve. If $x_J = 0$ then, $2J = O$ where, O is the point at infinity.

3.1.2 Elliptic Curve Domain Parameters

Apart from the curve parameters a and b, there are other parameters that must be agreed by both parties involved in secured and trusted communication using ECC. These are known as *domain parameters*. The domain parameters for prime fields and binary fields are described below. Generally, the protocols implementing the ECC specify the domain parameters to be used.

Domain parameters for EC over F_p

The domain parameters for Elliptic curve over F_p are p, a, b, G, n and h. p is the prime number defined for finite field F_p. a and b are the parameters defining the curve $y^2 \bmod p = (x^3 + ax + b) \bmod p$. G is the generator point (x_G, y_G), a point on the elliptic curve chosen for cryptographic operations. n is the order of the elliptic curve. The scalar for point multiplication is chosen as a number between 0 and $n - 1$. h is the cofactor where, $h = |E(F_p)|/n$. $|E(F_p)|$ is the number of points on an elliptic curve.

Domain parameters for EC over F_2^m

The domain parameters for elliptic curve over F_2^m are m, $f(x), a, b, G, n$ and h. m is an integer defined for finite field F_2^m. The elements of the finite field F_2^m are integers of length at most m bits. $f(x)$ is the irreducible polynomial of degree m used for elliptic curve operations. a and b are the parameters defining the curve $y^2 + xy = x^3 + ax^2 + b$. G is the generator point (x_G, y_G), a point on the elliptic curve chosen for cryptographic operations. n is the order of the elliptic curve. The scalar for point multiplication is chosen as a number between 0 and $n-1$. h is the cofactor where, $h = |E(F_2^m)|/n$. $|E(F_2^m)|$ is the number of points on an elliptic curve.

3.1.3 Elgamal-like Elliptic Curve Cryptosystem

In this section, we discuss ECC based on Elgamal. Suppose Alice wishes to send a message M to Bob. Initially, curve parameters p, a, b, G, n and h are agreed between Alice and Bob. First, she imbeds the value m onto the elliptic curve E, i.e., she represents the plaintext M as a point $P_m \in E$. Now she must encrypt P_m. Let d_B denote Bob's secret key and public key $Q = d_B G$. Alice first chooses a random integer k and sends Bob a pair of points (C_1, C_2) on E, where

$$C_1 = kG \tag{3.7a}$$

$$C_2 = P_m + kQ \tag{3.7b}$$

To decrypt the cipher text, Bob computes

$$\begin{aligned} C_2 - d_B C_1 &= P_m + kQ - d_B kG \\ &= P_m + kd_B G - d_B kG \\ &= P_m \end{aligned}$$

3.2 Cryptosystem for Large Message Encryption based on DLP

In 2002, Hwang et al. (HCH) [52] proposed an ElGamal-like asymmetric cryptosystem that allows a large message to be enciphered efficiently.

3.2.1 ElGamal Cryptosystem based on DLP

Let p be a prime, and g be a generator of Z_p. The private key x is an integer between 1 and $p-2$. Let $y = g^x \bmod p$. The public key for ElGamal encryption is the triplet (p, g, y). If taking discrete logarithms is as difficult as it is widely believed, releasing $y = g^x \bmod p$ does not reveal x. To encrypt a plaintext M, a random integer r relatively prime to $p-1$ is selected, and the following pair of values is computed as:

$$a = g^k \bmod p$$

$$b = My^k \bmod p$$

The cipher text C consists of the pair (a, b).

The decryption of the ciphertext $C = (a, b)$ in the ElGamal scheme, to retrieve the plaintext M, is simple as:

$$M = b/a^x \bmod p$$

3.2.2 HCH Cryptosystem

Let us assume Alice A wants to send a message M to Bob B. Their scheme works as follows:

1. **Key Generation:** B picks a large prime p and a primitive root g mod p. He also chooses a random exponent $\alpha \in Z_p$ and computes $\beta = g^{\alpha} \bmod p$. B's public key is (g, p, β) and the secret key is α.

2. **Encryption:** A breaks message M into n plaintexts $M_1, M_2, ..., M_n$, each $|log_2 P|$ bits large. Note $0 \leq M_i < p$. A then chooses two random exponents r_1 and r_2 with $1 < r_1, r_2 \leq p-1$ and computes $R_1, R_2, C_1, C_2, ..., C_n$ according as
   $R_1 = g^{r_1} \bmod p$,
   $R_2 = g^{r_2} \bmod p$,
   $C_i = M_i.(\beta^{r_1} \oplus (\beta^{r_2})^{2^i}) \bmod p$.
   Above, β^{r_1} and $(\beta^{r_2})^{2^i}$ are calculated in modulo p, and $\oplus$ is the bitwise exclusive-OR operation. The whole encrypted message is $(R_1, R_2, C_1, C_2, ..., C_n)$. A sends it to B through a public network.

3. **Decryption:** After receiving $(R_1, R_2, C_1, C_2, ..., C_n)$, B reconstructs the plaintexts M_i as
   $M_i = C_i.(R_1^{\alpha} \oplus (R_2^{\alpha})^{2^i})^{-1} \bmod p$.
   Above, R_1^{α} and $(R_2^{\alpha})^{2^i}$ are calculated in modulo p.

In 2004, Yuh-Dauh Lyuu et al. [53] in their cryptoanalysis, proved that HCH is inscure if the number of plaintexts n exceeds the order of 2 modulo q and the prime modulus p is chosen to be of the form $2^e q$, where e is a positive integer and q is a prime.

3.3 Proposed Schemes

Due to more overhead of IFP- and DLP-based cryptosystem for encrypting large message, two ECDLP-based schemes have been proposed in this section.

3.3.1 Large Message Encryption Scheme 1 (LMES1)

This proposed scheme is based on both the Diffie-Hellman distribution scheme and ElGamal cryptosystem. The Diffie-Hellman key distribution scheme is used to generate the key pair of public and secret keys for all users u_i for $i = 1, 2, ..., n$.

Each user u_i randomly selects secret key d_i and computes the corresponding public key $Q_i = d_i P$.

In this scheme, let Bob and Alice want to deliver a confidential large message M as per the following algorithm:

Initially, Bob breaks the plaintext $M(M_x, M_y)$ into t pieces $M_1, M_2, ..., M_t$ of length being 512 bits and convert them into points in EC.

***Key Generation (Alice)*:**

1. Select a random integer d from $[1, n-1]$.
2. Compute $Q = dP$.
3. A's public key is Q and private key is d.

***Encryption (Bob)*:**

1. Select two random numbers $(r_1, r_2) \in [1, n-1]$.
2. Compute B_1 and B_2 as follows:

$$B_1 = r_1 P \tag{3.8a}$$

$$B_2 = r_2 P \tag{3.8b}$$

   such that
   $S_{AB1} = r_1 Q = r_1 dP = (x_{s1}, y_{s1})$
   $S_{AB2} = r_2 Q = r_2 dP = (x_{s2}, y_{s2})$
3. if $x_{s1} = 0 \bmod P$ and $x_{s2} = 0 \bmod P$ then go to step 2.
4. Compute C_{xj} and $C_{yj}, j = 1, 2, ..., t$ as follows:

$$C_{xj} = M_{xj} * (x_{s1} \oplus x_{s2}^{2j}) \bmod n \tag{3.9a}$$

$$C_{yj} = M_{yj} * (y_{s1} \oplus y_{s2}^{2j}) \bmod n \tag{3.9b}$$

5. Send $(B_1, B_2, C_{xj}, C_{yj}), j = 1, 2, ..., t$ to Alice.

***Decryption (Alice)*:**

1. Alice receives $(B_1, B_2, C_{xj}, C_{yj}), j = 1, 2, ..., t$ and does the following to get $M = (M_x, M_y)$.

2. Compute S_{AB1} and S_{AB1} as follows:
   $S_{AB1} = dB_1 = dr_1P = (x_{s1}, y_{s1})$
   $S_{AB2} = dB_2 = dr_2P = (x_{s2}, y_{s2})$

3. Compute M_{xj} and M_{yj} as follows:
   $M_{xj} = C_{xj} * (x_{s1} \oplus x_{s2}^{2j})^{-1} \bmod n$
   $M_{yj} = C_{yj} * (y_{s1} \oplus y_{s2}^{2j})^{-1} \bmod n$

4. Find Message $M_j = (M_{xj}, M_{yj})$.

As per this proposed algorithm, the sender is required to select two random numbers r_1 and r_2. In comparison to $2t$ times in ElGamal-like ECC, this scheme needs four and two times scalar point multiplication for encryption and decryption respectively. However, for multiplication operation, it is same in both the cases. In case of the proposed scheme, it requires additional $4t$ EX-OR operations. While encrypting large messages, this proposed scheme is computationally faster as compared to ElGamal based EC cryptosystem, because the computational complexity depends on elliptic scalar point multiplication.

3.3.2 Large Message Encryption Scheme 2 (LMES2)

This scheme is based on both the Diffie-Hellman distribution scheme and ElGamal cryptosystem as before. Here, let Bob and Alice want to deliver a confidential large message M as per the following algorithm:

Initially, Bob breaks the plaintext $M(M_x, M_y)$ into t pieces $M_1, M_2, ..., M_t$ of length being 512 bits and convert them into points in EC.

***Key Generation (Alice)*:**

1. Select a random integer d from $[1, n-1]$.

2. Compute $Q = dP$.

3. Alice's public key is Q and private key is d.

***Encryption (Bob)*:**

1. Select a random number $r \in [1, n-1]$.

2. Compute K as follows:

$$K = rP \tag{3.10}$$

such that
$S_{AB} = rQ = rdP = (x_s, y_s)$

3. Compute C_{xj} and $C_{yj}, j = 1, 2, ..., t$ as follows:

$$C_{xj} = M_{xj}.(x_s + H(x_s, j)) \bmod n \tag{3.11a}$$

$$C_{yj} = M_{yj}.(y_s + H(y_s, j)) \bmod n \tag{3.11b}$$

where $H() : 0, 1^* \rightarrow Z_p$ is a cryptographic hash function.

4. Send $(K, C_{xj}, C_{yj}), j = 1, 2, ..., t$ to Alice.

***Decryption (Alice)*:**

1. Alice receives $(K, C_{xj}, C_{yj}), j = 1, 2, ..., t$ and does the following to get $M = (M_x, M_y)$.

2. Compute S_{AB} as follows:
$S_{AB} = dK = drP = (x_s, y_s)$

3. Compute M_{xj} and M_{yj} as follows:
$M_{xj} = C_{xj}/(x_s + H(x_s, j)) \bmod n$
$M_{yj} = C_{yj}/(y_s + H(y_s, j)) \bmod n$

4. Find Message $M_j = (M_{xj}, M_{yj})$.

3.3.3 Security Analysis of LMES1 and LMES2

Large Message Encryption Scheme 1

If an intruder knows some pairs of the plaintext and ciphertext (M_{xj}, C_{xj}) and (M_{yj}, C_{yj}), the intruder can obtain $(x_{s1} \oplus x_{s2}^{2j})$ and $(y_{s1} \oplus y_{s2}^{2j})$ by solving equations 3.9a and 3.9b. However, it is difficult for intruder to obtain $x_{s1}, x_{s2}, y_{s1}, y_{s2}, x_{s2}^{2j}$ and y_{s2}^{2j} from $(x_{s1} \oplus x_{s2}^{2j})$ mod n and $(y_{s1} \oplus y_{s2}^{2j})$ mod n. Since $(x_{s1} \oplus x_{s2}^{2j})$ and $(y_{s1} \oplus y_{s2}^{2j})$ are nonlinear, an illegal user can't acquire the pieces of plaintext even though he/she can find some (M_{xj}, C_{xj}) and (M_{yj}, C_{yj}) by solving 3.9a and 3.9b. Hence the proposed scheme is secure against a chosen plaintext attacks.

Large Message Encryption Scheme 2

In this scheme, for an adversary it is infeasible to compute either d or r from S_{AB}. Suppose, the adversary knows $t-1$ parts of the plaintext message: $M_{x1}, .., M_{xi-1}$, $M_{xi+1}, ..., M_{xt}$ and $M_{y1}, .., M_{yi-1}$, $M_{yi+1}, ..., M_{yt}$. Then, the adversary can only derive $x_s + H(x_s, j) = C_{xj}/M_{yj}$ for $j \neq i$, but this doesn't allow the adversary to compute $x_s + H(x_s, i)$ and $y_s + H(y_s, i)$. Hence, the adversary still can't compute M.

3.4 Results and Discussion

The proposed schemes have been implemented using Java Card Toolkit 2-1-2 and tested in Java Card simulator— *jcwde* (Java Card Workstation Development Environment) and the encryption time comparison between LMES1, LMES2 and ElGamal-like ECC is shown in Figure 3.6. It is seen from Figure 3.6 that the rate of growth of computing time against message size of LMES1 and LMES2 are much less than the ElGamal-like ECC. Also, it is observed that LMES2 is faster than that of LMES1.

The number of operations used in different schemes are compared and shown in Table 3.1.

It can be observed from Table 3.1 that the number of scalar point multiplication operations in LMES1 and LMES2 are less than that of ElGamal like-ECC. Other operations used in LMES1 and LMES2 are least significant than that of

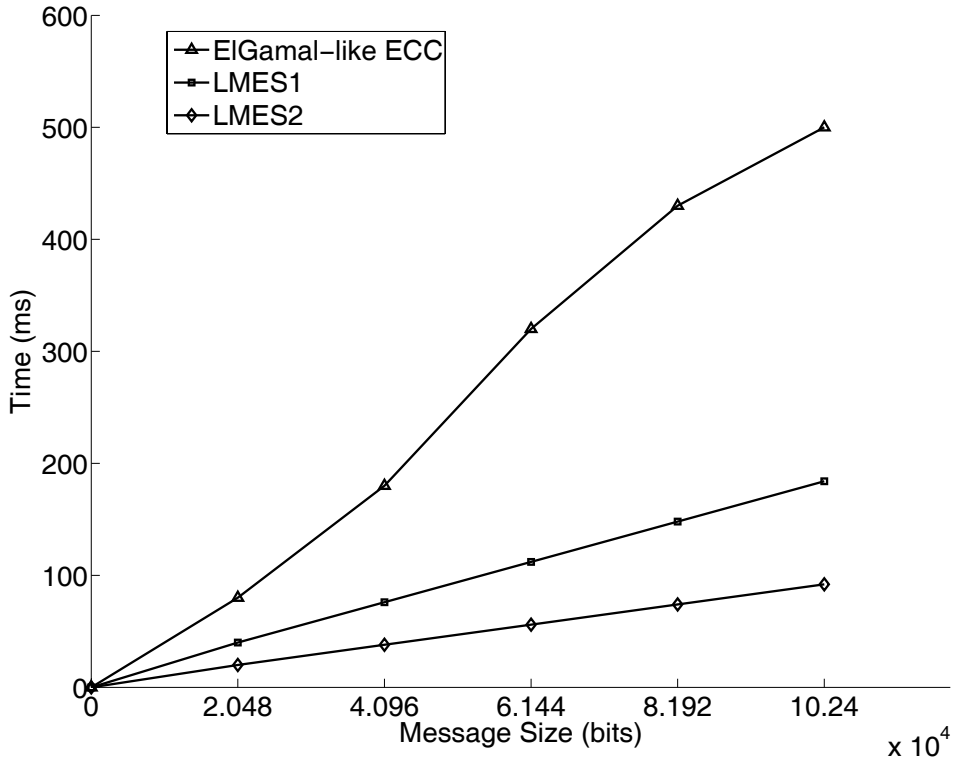


Figure 3.6: Encryption Times of ElGamal-like ECC, LMES1, and LMES2

scalar point multiplication operations.Because of this, LMES1 and LMES2 are computationally faster than ElGamal like-ECC.

Table 3.1: Comparison among ElGamal-like ECC, LMES1, and LMES2

Schemes	Encryption	Decryption
ElGamal-ECC	t scalar point multiplications	t scalar point multiplications
LMES1	4 scalar point multiplications $2t$ EX-OR operations $2t$ modular multiplications 2 random number generations	2 scalar point multiplications $2t$ EX-OR operations $2t$ modular divisions
LMES2	2 scalar point multiplications $2t$ modular multiplications $2t$ hash operations 1 random number generations	1 scalar point multiplication $2t$ modular divisions $2t$ hash operations

Chapter 4

Digital and Blind Signature Schemes

Chapter 4

Digital and Blind Signature Schemes

Establishing a framework for the authentication of computer-based information in today's commercial era requires awareness with concepts and specialized skills from computer security fields. A digital signature scheme offers a technique for Signer to sign document electronically in a secure and efficient manner using his private key so that the signatures can later be verified by anyone else by using public key of Signer. In this way, it will be difficult to forge the signatures. In order to ensure the security and authenticity of the documents, the Information Technology Act, 2000 of India provides rules for the use of Digital Signatures on the documents submitted in electronic form. The idea of public key cryptosystem was first defined in the path-breaking paper of Diffie and Hellman, and later digital signature algorithm (DSA) based on a number theoretic assumption was given by Rivest, Shamir and Adleman [38]. Digital signature's formal definition was first outlined by Goldwasser, Micali, and Rivest [54].

Following are some common reasons for applying a digital signature to communications.

- **Signer Authentication:** If a public and private key pair is associated with an identified Signer, the digital signature attributes the message to the Signer. The digital signature cannot be forged unless the Signer loses control of the private key (compromise of the private key), e.g., by means of revealing it or losing the media or device in which it is contained.

- **Message Authentication:** The digital signature also identifies the signed message, generally with a much greater certainty and precision than paper signatures. Verification reveals any tampering, since the comparison of the hash results (one made at signing and the other made at verifying) shows whether the message is the same as when signed.

- **Affirmative Act:** Creating a digital signature requires the Signer to use the Signer's private key. This act can perform the ceremonial function of alerting the Signer to the fact that the Signer is consummating a transaction with legal consequences.

- **Efficiency:** The processes of creating and verifying a digital signature provide a high level of assurance that the digital signature is genuinely the Signer's. As for modern electronic data interchange (EDI), the creation and verification processes are capable of being completely automatic (sometimes referred to as machinable), and the human interaction is required in an exception case only. Compared to paper methods such as checking specimen signature cards— methods so tedious and labor-intensive that they are actually, rarely used in practice. Digital signatures yield a high degree of assurance without adding greatly to the resources required for processing.

An interesting variant of the basic digital signature scheme (DSS) is the blind digital signature. The concept of a blind signature scheme (BSS) was introduced by Chaum [55] to enable spender anonymity in ECS. This procedure can be well explained with an example taken from the familiar world of paper documents. The paper analog of a blind signature can be implemented with carbon paper lined envelopes. Putting a signature on the outside of such an envelope leaves a carbon copy of the signature on a slip of paper within the envelope. Such signatures require that a Signer be able to sign a document without knowing its contents. Moreover, in case the Signer see the document/signature pair, still then one will not be able to determine when or for whom it has been signed (but the verification of the signature can be made). This intuitively refers to signing a document with your eyes closed. Even if you see the document and signature later on, you can verify that the signature, but you will probably have great difficulty in recollecting

when or for whom you signed the original document. At first this concept seems a little strange (why would you want to sign something without seeing it?). It turns out that, when applied properly, this notion has some very nice applications in situations where anonymity is a big issue. Two such applications are online voting and electronic cash. When you submit an online vote, you might like for that vote to be anonymous so no one can tell whom you voted for. Similarly with electronic cash, you might not want someone else to know who you are when you spend it. This is similar to normal paper cash when you make a purchase, the vendor more or less has no idea who you are, but he can probably tell whether the money you gave him is legitimate. Specifically, in this electronic cash scenario, a document corresponds to an electronic coin or note, and signed it.

In this chapter, Section 4.1 describes a high-level picture of the Digital Signature and Blind Signature based on RSA public key cryptosystem. In Section 4.2, application of Blind Signatures are explained. In Section 4.3, proposed BSS have been explained. Properties of proposed BSS like correctness blindness, untracability and unforgebility have been discussed in Section 4.4. Finally, results and discussion of proposed BSS are done in Section 4.5.

4.1 Signature Schemes Based on RSA

The digital and blinds signature schemes based on RSA public key cryptosystem are explained below.

4.1.1 Digital Signature

A digital signature scheme (DSS) typically consists of three algorithms:

- A ***key generation*** algorithm that selects a private key uniformly at random from a set of possible private keys. The algorithm outputs the private key and a corresponding public key.
- A ***signing*** algorithm which, given a message and a private key, produces a signature.

- A ***signature verifying*** algorithm which given a message, public key and a signature, either accepts or rejects.

Key Generation Algorithm

Let $n = pq$ where p and q are two large primes and let e be chosen such that $GCD(\phi(n), e) = 1,\ (1 < e < \phi(n))$. Moreover, let d be such that $de = 1 \bmod \phi(n)$. We assume that the Signer's public key is (n, e) and the private key is (p, q, d). It is proved that obtaining a private key from the public key is very difficult unless you know the factorization of n [38].

Signing Algorithm

The signature can be computed easily by a Signer who knows the message and secret key d as follows:

$$s = m^d \mod n \tag{4.1}$$

Verifying Algorithm

It is easy to verify that (m, s) is valid by checking if the following equality holds:

$$m = s^e \mod n \tag{4.2}$$

where equality follows because $de = 1 \bmod \phi(n)$. Now we will discuss the Chaum's Blind Signature.

4.1.2 Blind Signature

Key Generation and parameters are same as Digital Signature. Let a Requester sends a message $M \in [0, n-1]$ to be signed by a Signer using Chaum's BSS. The different phases are explained below in brief.

Blinding Phase

The Requester picks a blinding factor r, which is a random integer between 0 and n, and computes the value:

$$m' = mr^e \mod n \tag{4.3}$$

The Requester sends m' to the Signer. m' is the message to be signed by the Signer as in case of general signature without knowing the original message m.

Signing Phase

The Signer signs the message m' using his/her private key d as below:

$$s' = (m')^d \mod n \tag{4.4}$$

The Signer returns s' to the Requester as the blind signature.

Extraction Phase

The Requester after receiving the s', he/she extracts the signature s as follows:

$$\begin{aligned} s &= s'/r \mod n \\ &= (m')^d/r \mod n \\ &= (mr^e)^d/r \mod n \\ &= m^d r^{ed}/r \mod n \\ &= m^d \mod n \end{aligned} \tag{4.5}$$

So the Requester finds the actual signature of m as (m, s) which satisfies the Eqn. (4.5). Figure 4.1 depicts the Chaum's BSS in terms of message communicated between the Requester and the Signer.

The most important thing to observe about this protocol is that Signer has issued the signature without ever seeing the actual message. This happens because the blinding factor r^e was multiplied to the message, and as a result the final message just looked like a random element of Z_n to Signer. Later, after the signature is issued, and Requester divides out by r (to remove the blinding factor), the resulting message-signature pair is unrecognizable to Signer. In fact, if Signer later sees this message, she can easily verify that the signature is indeed hers, but she will be critically limited in accurately determining when for whom she signed the message. At best, she may be able to make a random guess from among the signatures she has issued, but she cannot do any better.

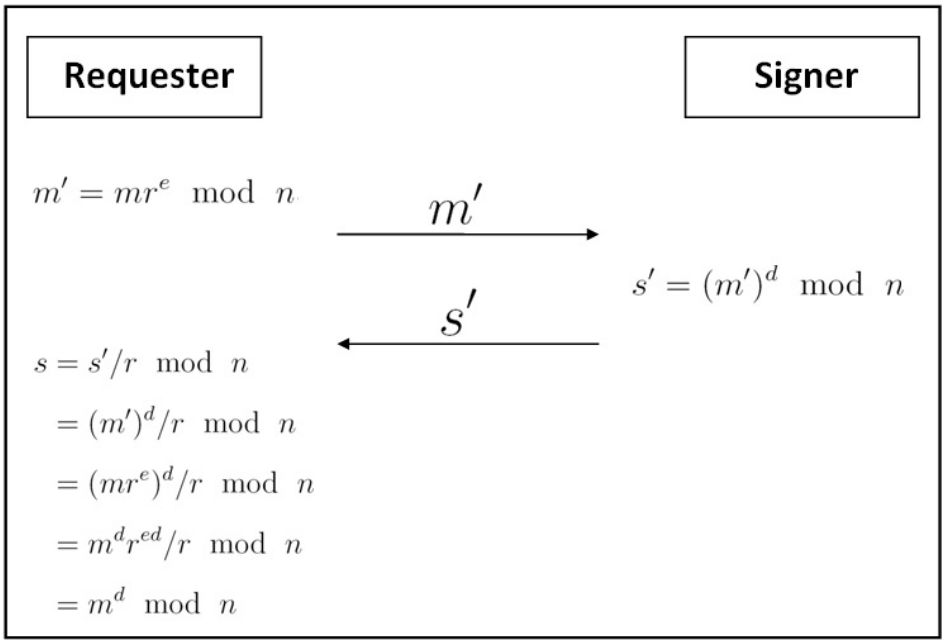


Figure 4.1: Chaum' s Blind Signature Scheme based on RSA

4.2 Applications of BSS

In this section, applications of BSS in digital cash and online voting have been discussed.

4.2.1 Digital Cash

We now outline how to apply the blind digital signature idea to digital cash [119]. We give a basic protocol due to Chaum [55] for a digital cash transaction. We describe a very simple version of the protocol primarily to give the main ideas. As a result, there are some minor flaws in it, but standard techniques exist to amend these flaws. There are several more comprehensive works on electronic cash which discuss this, as well as other well-known ECS [120,121]. In the protocol we present, we assume that there is a student who wants to buy a Book, which costs INR 250 from the fictitious online vendor e-Bay. Furthermore, Student and e-Bay both use the same bank, which we call the Bank. The entire transaction protocol is broken up into three stages— Withdrawal, Spending, and Deposit.

Withdrawal

1. Student creates a piece of digital currency C. This currency consists of a string of bits which specify certain information such as a large random *serial number* and the rupees amount (which is INR 250 in this case).

2. Student now takes C, and gets the Bank to blindly sign it, i.e., Student and the Bank engage in a blind signature protocol, at the end of which Student possesses a valid signature on the currency.

3. Upon successful completion of the protocol, the Bank deducts INR 250 from Student's Bank account.

Spending

1. Student requests a copy of the book he wants from e-Bay. Student takes this valid INR 250, C, along with the bank's signature on that Rupees, and gives it to e-Bay.

2. e-Bay checks that C is valid by verifying the signature on it with the Bank's public key. If the signature is invalid, e-Bay immediately ends the protocol.

Deposit

1. e-Bay now takes the currency C, and the Bank's signature on C, and gives it to the Bank.

2. The Bank verifies that the signature on the Rupees is indeed valid, i.e., the currency was indeed signed by the Bank. If it is valid, the bank needs to check that the coin has not yet been spent. If the signature on the coin is valid and the currency has not yet been spent, then the Bank credits e-Bay's bank account by INR 250.

3. If all checks out, then e-Bay gives Student the book she requested.

Thus we have completed a transaction. Observe that Student's identity remains anonymous to both the Bank and to e-Bay. This happens because a blind

signature was issued on the currency. Therefore, once Student has the signed currency, the vendor certainly will not be able to tell that Student is the owner of the currency. Moreover, when the vendor gives the money to the Bank for deposit, the Bank will not be able to link the currency it just received to Student, because it blindly signed the message, and therefore, it had no idea what the contents were.

4.2.2 Online Voting

We give an application of the traditional blind signature protocol to electronic voting. We now suppose that we have two entities— a voter Alice, and a Central Tabulating Authority (CTA). We assume that the votes are of the form *Yes* or *No*. The protocol we give is adapted from the one presented by Schneier [122]. The voting protocol is divided into two stages— *registration* and *voting*.

Registration

1. Alice creates two electronic ballots B_1 and B_2 which consist of a serial number, and some other relevant information for voting. In addition, B_1 contains the vote *Yes* and B_2 contains the vote *No*.

2. Alice then takes the ballots B_1, B_2 and blinds them. It sends the blind versions to the CTA.

3. The CTA checks its database to make sure that Alice has not voted before. If this is the case, then it signs the blinded ballots and it gives them back to Alice.

Voting

1. Alice unblinds the messages, and now has two sets of valid votes signed by the CTA.

2. Alice picks one vote (*Yes* or *No*) she wants to choose, along with the CTA's signature on that vote, and encrypts it with the CTA's public key.

3. Alice sends in her vote.

4. The CTA decrypts the votes, checks that the signature is valid, and checks its database to make sure that the serial number on the vote has not been used before (this prevents Alice from trying to vote more than once). If this checks out then the CTA tabulates the vote, saves the serial number and records it in the database. At the end of the election, the CTA publishes the results of the election, as well as every serial number and its associated vote.

4.3 Proposed Blind Signature Schemes

In this section, before describing four proposed BSS and its application to offline digital cash, first we will describe the variation of DSS based on ECC.

Initially, the curve parameters (p, a, b, G, n, h) must be agreed upon by Signer and Receiver which is explained in Section 3.1.2. Signer must have a key pair suitable for elliptic curve cryptography, consisting of a private key d_B (randomly selected in the interval $[1, n-1]$) and a public key Q, where $Q = d_B G$.

4.3.1 Digital Signature Schemes based on ECDLP

1. Based on Nyberg-Ruppel Scheme (BNRS)

When a Signer wants to send a signed message m to Receiver, he/she must generate a digital signature (R, s) as follows:

Message m is converted into a point M on EC.

Select $k \in [1, n-1]$ and generate R, r and s as:

$$R = M + kG \quad \text{where,} \quad R = (x_r, y_r)$$
$$r = x_r \mod n$$
$$s = (rd_B + k) \mod n$$

After receiving (R, s) from Signer, the Receiver can verify the correctness of the signature on the message by using following equation:

$$M = rQ + R - sG \tag{4.6}$$

Correctness

The verifier can verify the pair (R, s) and message m by using the above equation. The correctness of the equation is as follows:

$$\begin{aligned} & s = (rd_B + k) \bmod n \\ \Rightarrow\ & sG = rd_BG + kG \\ \Rightarrow\ & sG = rQ + kG \\ \Rightarrow\ & sG = rQ + R - M \\ \Rightarrow\ & M = rQ + R - sG \end{aligned}$$

2. Based on Variation of DSA (BDSA1)

When a Signer wants to send a signed message m to Receiver, a digital signature (R, s) is generated as follows:

Select $k \in [1, n-1]$ and generate R, r and s as:

$$R = kG \text{ where, } R = (x_r, y_r)$$

$$r = x_r \bmod n \text{ and } r \neq 0 \tag{4.7}$$

$$s = (km + rd_B) \bmod n \text{ and } s \neq 0 \tag{4.8}$$

After receiving (R, s) from the Signer, the Receiver can verify the correctness of the signature on the message by using following equation:

$$mR = sG - rQ \tag{4.9}$$

Correctness

The verifier can verify the pair (R, s) and message m by using the above equation. Also the correctness of the equation is as follows:

$$\begin{aligned} & s = (km + rd_B) \bmod n \\ \Rightarrow\ & sG = kmG + rd_BG \\ \Rightarrow\ & sG = Rm + rQ \\ \Rightarrow\ & mR = sG - rQ \end{aligned}$$

3. Based on Variation of DSA (BDSA2)

When a Signer wants to send a signed message m to Receiver, a digital signature (R, s) is generated as follows:

Select $k \in [1, n-1]$ and generate R, r and s as:

$$R = kG \text{ where, } R = (x_r, y_r)$$

$$r = x_r \bmod n \text{ and } r \neq 0 \tag{4.10}$$

$$s = (md_B - rk) \bmod n \text{ and } s \neq 0 \tag{4.11}$$

After receiving (R, s) from the Signer, the Receiver can verify the correctness of the signature on the message by using following equation:

$$mQ = sG + rR \tag{4.12}$$

Correctness

The verifier can verify the pair (R, s) and message m by using the above equation. Also the correctness of the equation is as follows:

$$\begin{aligned} & s = (md_B - rk) \bmod n \\ & \Rightarrow sG = md_BG - rkG \\ & \Rightarrow sG = mQ - rR \\ & \Rightarrow mQ = sG + rR \end{aligned}$$

4. Based on Variation of DSA (BDSA3)

When a Signer wants to send a signed message m to Receiver, a digital signature (R, s) is generated as follows:

Select $k \in [1, n-1]$ and generate R, r and s as:

$$R = kG \text{ where, } R = (x_r, y_r)$$

$$r = x_r \bmod n \text{ and } r \neq 0 \tag{4.13}$$

$$s = (mrd_B - k) \bmod n \text{ and } s \neq 0 \tag{4.14}$$

After receiving (R, s) from the Signer, the Receiver can verify the correctness of the signature on the message by using following equation:

$$sG = mrQ - R \tag{4.15}$$

Correctness

The verifier can verify the pair (R, s) and message m by using the above equation. Also the correctness of the equation is as follows:

$$\begin{aligned} & s = (mrd_B - k) \bmod n \\ & \Rightarrow sG = mrd_B G - kG \\ & \Rightarrow sG = mrQ - R \end{aligned}$$

4.3.2 Blind Signature Schemes based on ECDLP

In the proposed schemes, the Requester requests signature from the Signer, and the Signer issues blind signature to the Requester without knowing the content of the message. The protocol consists of following five phases:

(a) Initialization phase

(b) Blinding phase

(c) Signing phase

(d) Extraction phase

(e) Verifying phase

In the initialization phase, the system's parameter is defined, and the Signer publishes the necessary information and sends partially blind signature to Requester. To obtain a signature, a Requester submits an encrypted version (blinds the message) of the message to the Signer in the requesting phase. In the signing phase, the Signer computes the blind signature of the message, and then sends the result back to the Requester. In the extraction phase, the Requester extracts the signature from the blind signature he receives from the signer. Lastly, anyone can verify the legitimacy of the digital signature in the verifying phase. The details of proposed schemes are presented as follows.

1. Based on BNRS (BSSNRS)

The Initialization Phase

The Signer randomly chooses $\tilde{k}$, and calculates $\tilde{r}$ as follows:

$$\begin{aligned}&\tilde{R} = \tilde{k}G\\&\tilde{R} = (\tilde{x}_r, \tilde{y}_r)\\&\tilde{r} = \tilde{x}_r \bmod n \text{ and } \tilde{r} \neq 0\end{aligned}$$

The Signer sends $\tilde{R}$ to Requester.

The Blinding Phase

After receiving $\tilde{R}$, Requester randomly selects integers $\alpha, \beta \in [1, n-1]$ and computes R as:

$$\begin{aligned}&R = M + \alpha G + \beta\tilde{R}\\&\Rightarrow \beta\tilde{R} + \alpha G = R - M\\&r = x_r \bmod n\end{aligned} \tag{4.16}$$

After calculating r, the Requester blinds the message m as follows:

$$\tilde{m} = r\beta^{-1} \bmod n \tag{4.17}$$

The Requester sends the blind messages $\tilde{m}$ to Signer for signature.

The Signing Phase

In this phase, the Signer computes blind signature $\tilde{s}$ by using received blind message $\tilde{m}$ as follows:

$$\tilde{s} = (\tilde{m}d_B + \tilde{k}) \bmod n \tag{4.18}$$

Then the Signer sends the blind signatures $\tilde{s}$ to the Requester.

The Extraction Phase

After receiving the blind signature $\tilde{s}$, the Requester extracts the actual signature as follows:

$$s = (\tilde{s}\beta + \alpha) \bmod n \tag{4.19}$$

The pair (r, s) are the valid digital signature of message m.

The Verifying Phase

The legitimacy of the digital signature (R, r, s) of message m is as follows:

$$M = rQ + R - sG \tag{4.20}$$

The proposed scheme is diagrammatically shown in Fig. 4.2.

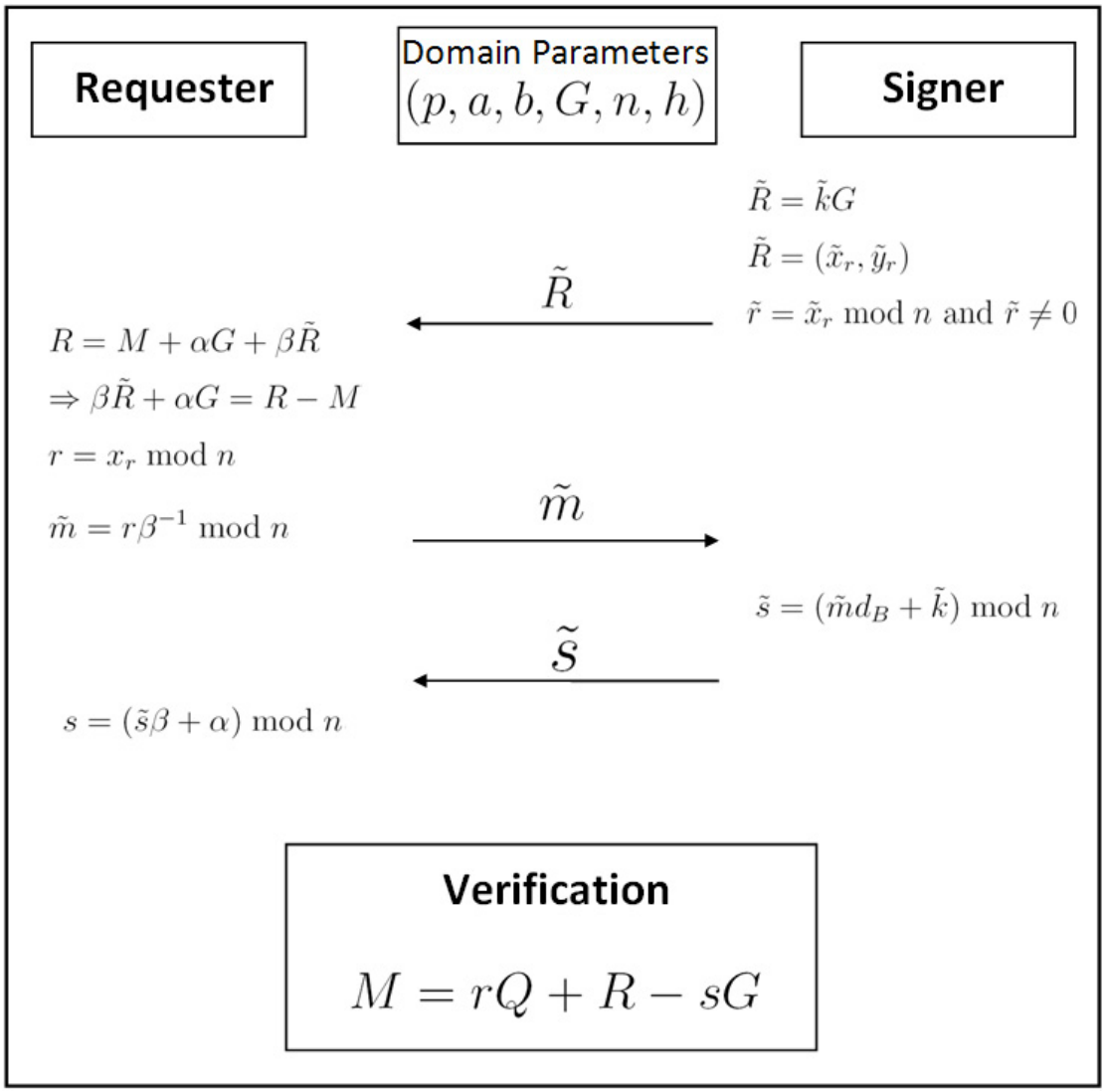


Figure 4.2: Proposed blind signature scheme: BNRS

2. Based on BDSA1 (BSSDSA1)

The Initialization Phase

The Signer randomly chooses $\tilde{k}$, and calculates $\tilde{r}$ as follows:

$$\begin{aligned}\tilde{R} &= \tilde{k}G \\ \tilde{R} &= (\tilde{x}_r, \tilde{y}_r) \\ \tilde{r} &= \tilde{x}_r \bmod n \text{ and } \tilde{r} \neq 0\end{aligned}$$

The Signer sends $\tilde{R}$ to Requester.

The Blinding Phase

After receiving $\tilde{R}$, Requester randomly selects integers α, β and computes R as:

$$\begin{aligned}R &= \alpha\tilde{R} + \beta G \\ r &= x_r \bmod n\end{aligned} \tag{4.21}$$

After calculating r, the Requester blinds the message m as follows:

$$\tilde{m} = \alpha m \tilde{r} r^{-1} \bmod n \tag{4.22}$$

The Requester sends the blind messages $\tilde{m}$ to Signer for signature.

The Signing Phase

In this phase, the Signer computes blind signature $\tilde{s}$ by using received blind message $\tilde{m}$ as follows:

$$\tilde{s} = (\tilde{k}\tilde{m} + \tilde{r}d_B) \bmod n \tag{4.23}$$

Then the Signer sends the blind signatures $\tilde{s}$ to the Requester.

The Extraction Phase

After receiving the blind signature $\tilde{s}$, the Requester extracts the actual signature as follows:

$$s = (\tilde{s}r\tilde{r}^{-1} + \beta m) \bmod n \tag{4.24}$$

The pair (r, s) are the valid digital signature of message m.

The Verifying Phase

The legitimacy of the digital signature (R, r, s) of message m is as follows:

$$mR = sG - rQ \tag{4.25}$$

The proposed scheme is diagrammatically shown in Fig. 4.3.

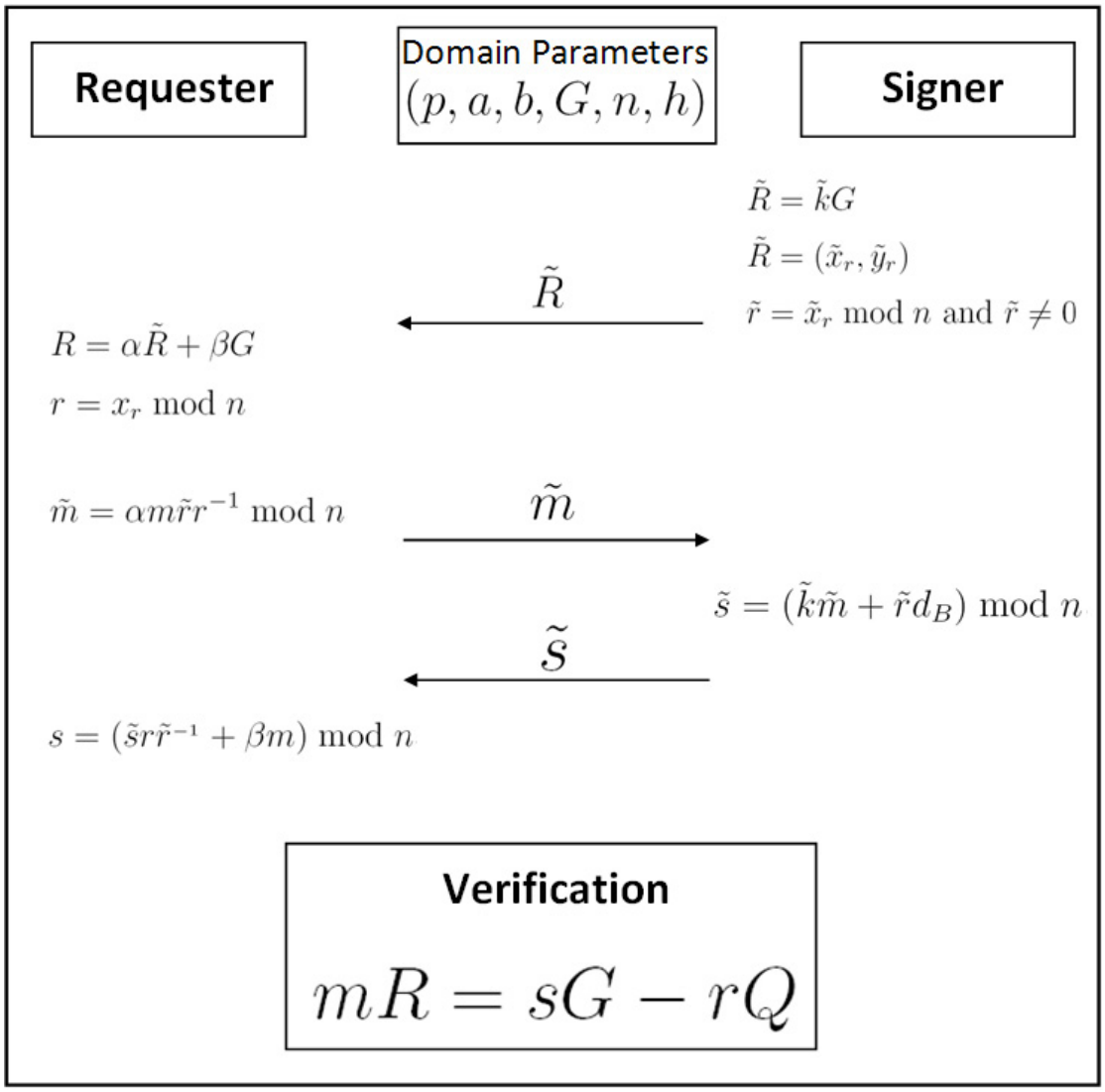


Figure 4.3: Proposed blind signature scheme: BSSDSA1

3. Based on BDSA2 (BSSDSA2)

The Initialization Phase

In the initialization phase the Signer randomly chooses $\tilde{k}_1, \tilde{k}_2, l_1$ and l_2, and calculates $\tilde{r}_1$ and $\tilde{r}_2$ as follows:

$$\begin{aligned} \tilde{R}_1 &= \tilde{k}_1 G \\ \tilde{R}_2 &= \tilde{k}_2 G \end{aligned} \tag{4.26}$$

where $\tilde{R}_1 = (\tilde{x}_{r_1}, \tilde{y}_{r_1})$ and $\tilde{R}_2 = (\tilde{x}_{r_2}, \tilde{y}_{r_2})$

$$\begin{aligned} \tilde{r}_1 &= \tilde{x}_{r_1} \bmod n \text{ and } \tilde{r}_1 \neq 0 \\ \tilde{r}_2 &= \tilde{x}_{r_2} \bmod n \text{ and } \tilde{r}_2 \neq 0 \end{aligned} \tag{4.27}$$

The Signer sends $(\tilde{R}_1, \tilde{R}_2, l_1, l_2)$ to Requester.

The Blinding Phase

After receiving $(\tilde{R}_1, \tilde{R}_2, l_1, l_2)$, the Requester should request for signature by computing the values as follows. The Requester randomly selects four integers α, β, γ, and η such that α is relatively prime to β, i.e., $GCD(\alpha, \beta) = 1$. According to Extended Eculid's algorithm there exist two integers i and j such that $(i\gamma + j\eta) \bmod n = 1$. The Signer's secret values are $(\alpha, \beta, \gamma, \eta, i, j)$. The Requester computes R_1 and R_2 as follows:

$$\begin{aligned} R_1 &= \gamma\alpha l_1 \tilde{R}_1 \quad \text{where,} \quad R_1 = (x_{r_1}, y_{r_1}) \\ R_2 &= \eta\beta l_2 \tilde{R}_2 \quad \text{where,} \quad R_2 = (x_{r_2}, y_{r_2}) \\ r_1 &= x_{r_1} \bmod n \\ r_2 &= x_{r_2} \bmod n \\ r &= (r_1 * r_2) \bmod n \\ R &= R_1 + R_2 \end{aligned} \tag{4.28}$$

After calculating r_1 and r_2, the Requester blinds the message m as follows:

$$\begin{aligned} \tilde{m}_1 &= im\tilde{r}_1 r_1^{-1} r_2^{-1} \alpha^{-1} \bmod n \\ \tilde{m}_2 &= jm\tilde{r}_2 r_1^{-1} r_2^{-1} \beta^{-1} \bmod n \end{aligned} \tag{4.29}$$

The Requester sends the blind messages $\tilde{m}_1$ and $\tilde{m}_2$ to Signer for signature.

The Signing Phase

In this phase, the Signer computes blind signatures $\tilde{s}_1$ and $\tilde{s}_2$ by using received blind messages $\tilde{m}_1$ and $\tilde{m}_2$ as follows:

$$\begin{aligned} \tilde{s}_1 &= (d_B \tilde{m}_1 - \tilde{r}_1 \tilde{k}_1 l_1) \bmod n \\ \tilde{s}_2 &= (d_B \tilde{m}_2 - \tilde{r}_2 \tilde{k}_2 l_2) \bmod n \end{aligned} \tag{4.30}$$

Then the Signer sends the blind signatures $\tilde{s}_1$ and $\tilde{s}_2$ to the Requester.

The Extraction Phase

Requester should do the followings to recover the real signature s after receiving the blinded signatures $\tilde{s}_1$ and $\tilde{s}_2$ from the Signer.

$$\begin{aligned} s_1 &= \tilde{s}_1 \tilde{r}_1^{-1} r_1 r_2 \gamma \alpha \bmod n \\ s_2 &= \tilde{s}_2 \tilde{r}_2^{-1} r_1 r_2 \eta \beta \bmod n \\ s &= (s_1 + s_2) \bmod n \end{aligned} \tag{4.31}$$

$$\begin{aligned} R &= R_1 + R_2 \\ r &= (r_1 . r_2) \bmod n \end{aligned} \tag{4.32}$$

The pair (r, s) is the valid digital signature of message m.

The Verifying Phase

The legitimacy of the digital signature (R, r, s) of message m is as follows:

$$mQ = sG + rR \tag{4.33}$$

The proposed scheme is diagrammatically shown in Fig. 4.4 .

4. Based on BDSA3 (BSSDSA3)

The Initialization Phase

In the initialization phase the Signer randomly chooses $\tilde{k}_1, \tilde{k}_2, l_1$ and l_2, and calculates $\tilde{r}_1$ and $\tilde{r}_2$ as follows:

$$\begin{aligned} \tilde{R}_1 &= \tilde{k}_1 G \\ \tilde{R}_2 &= \tilde{k}_2 G \end{aligned} \tag{4.34}$$

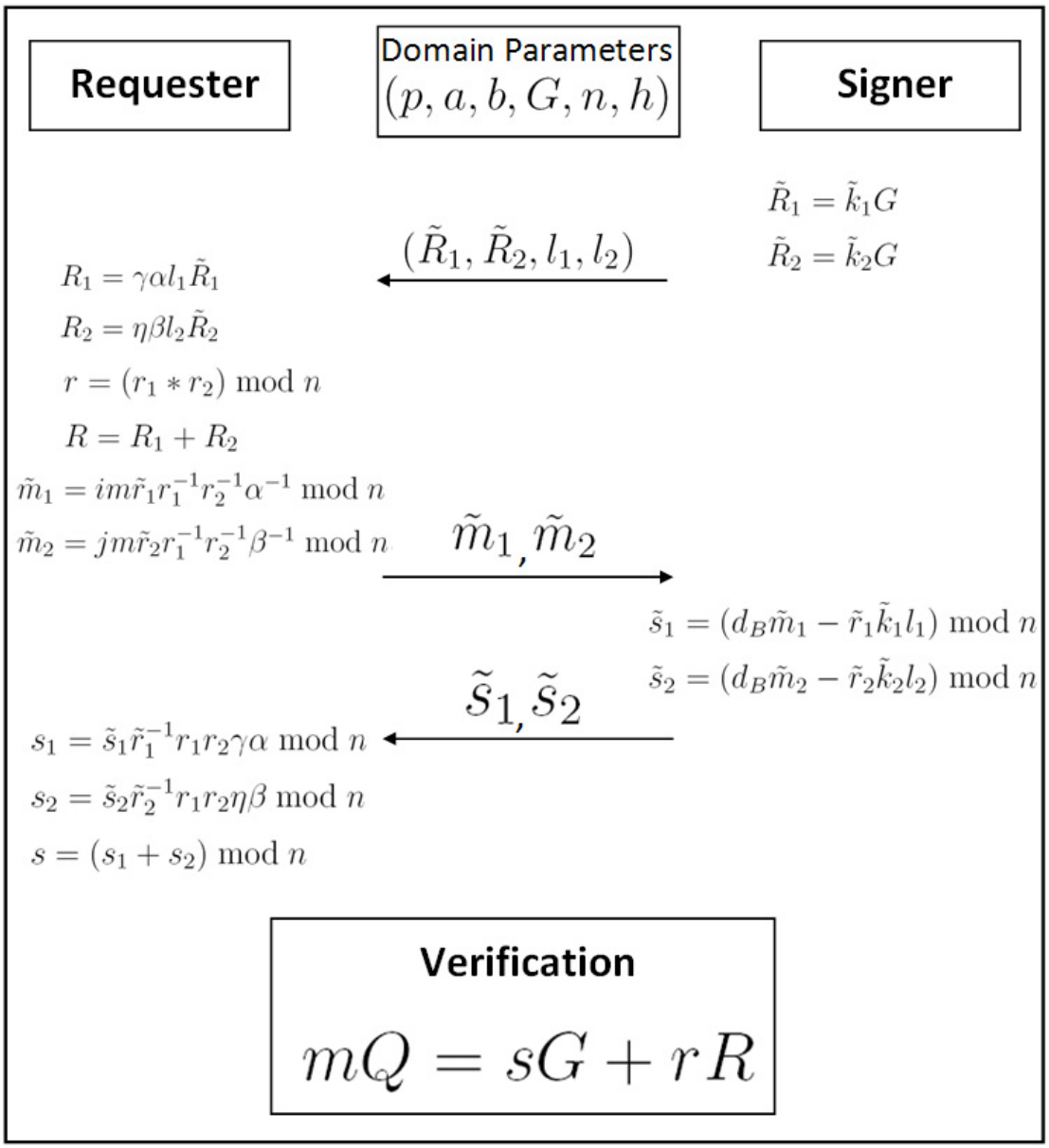


Figure 4.4: Proposed blind signature scheme: BSSDSA2

where $\tilde{R}_1 = (\tilde{x}_{r_1}, \tilde{y}_{r_1})$ and $\tilde{R}_2 = (\tilde{x}_{r_2}, \tilde{y}_{r_2})$

$$\begin{aligned} \tilde{r}_1 &= \tilde{x}_{r_1} \bmod n \text{ and } \tilde{r}_1 \neq 0 \\ \tilde{r}_2 &= \tilde{x}_{r_2} \bmod n \text{ and } \tilde{r}_2 \neq 0 \end{aligned} \tag{4.35}$$

The Signer sends $(\tilde{R}_1, \tilde{R}_2, l_1, l_2)$ to Requester.

The Blinding Phase

After receiving $(\tilde{R}_1, \tilde{R}_2, l_1, l_2)$, the Requester should request for signature by computing the values as follows. The Requester randomly selects four integers $\alpha, \beta, \gamma,$

and η such that α is relatively prime to β, i.e., $GCD(\alpha, \beta) = 1$. According to Extended Eculid's algorithm there exist two integers i and j such that $(i\gamma + j\eta) \bmod n = 1$. The Signer's secret values are $(\alpha, \beta, \gamma, \eta, i, j)$. The Requester computes R_1 and R_2 as follows:

$$\begin{aligned} R_1 &= \gamma\alpha l_1 \tilde{R}_1 \quad \text{where,} \quad R_1 = (x_{r_1}, y_{r_1}) \\ R_2 &= \eta\beta l_2 \tilde{R}_2 \quad \text{where,} \quad R_2 = (x_{r_2}, y_{r_2}) \\ r_1 &= x_{r_1} \bmod n \\ r_2 &= x_{r_2} \bmod n \\ r &= (r_1 * r_2) \bmod n \\ R &= R_1 + R_2 \end{aligned} \tag{4.36}$$

After calculating r_1 and r_2, the Requester blinds the message m as follows:

$$\begin{aligned} \tilde{m}_1 &= im\tilde{r}_1^{-1} r_1 r_2 \alpha^{-1} \bmod n \\ \tilde{m}_2 &= jm\tilde{r}_2^{-1} r_1 r_2 \beta^{-1} \bmod n \end{aligned} \tag{4.37}$$

The Requester sends the blind messages $\tilde{m}_1$ and $\tilde{m}_2$ to Signer for signature.

The Signing Phase

In this phase, the Signer computes blind signatures $\tilde{s}_1$ and $\tilde{s}_2$ by using received blind messages $\tilde{m}_1$ and $\tilde{m}_2$ as follows:

$$\begin{aligned} \tilde{s}_1 &= (d_B \tilde{m}_1 \tilde{r}_1 - \tilde{k}_1 l_1) \bmod n \\ \tilde{s}_2 &= (d_B \tilde{m}_2 \tilde{r}_2 - \tilde{k}_2 l_2) \bmod n \end{aligned} \tag{4.38}$$

Then the Signer sends the blind signatures $\tilde{s}_1$ and $\tilde{s}_2$ to the Requester.

The Extraction Phase

Requester should do the followings to recover the real signature s after receiving the blinded signatures $\tilde{s}_1$ and $\tilde{s}_2$ from the Signer.

$$\begin{aligned} s_1 &= \tilde{s}_1 \gamma\alpha \bmod n \\ s_2 &= \tilde{s}_2 \eta\beta \bmod n \\ s &= (s_1 + s_2) \bmod n \end{aligned} \tag{4.39}$$

$$\begin{aligned} R &= R_1 + R_2 \\ r &= (r_1 . r_2) \bmod n \end{aligned} \tag{4.40}$$

The pair (r, s) is the valid digital signature of message m.

The Verifying Phase

The legitimacy of the digital signature (R, r, s) of message m is as follows:

$$mrQ = R + sG \tag{4.41}$$

The proposed scheme is diagrammatically shown in Fig. 4.5.

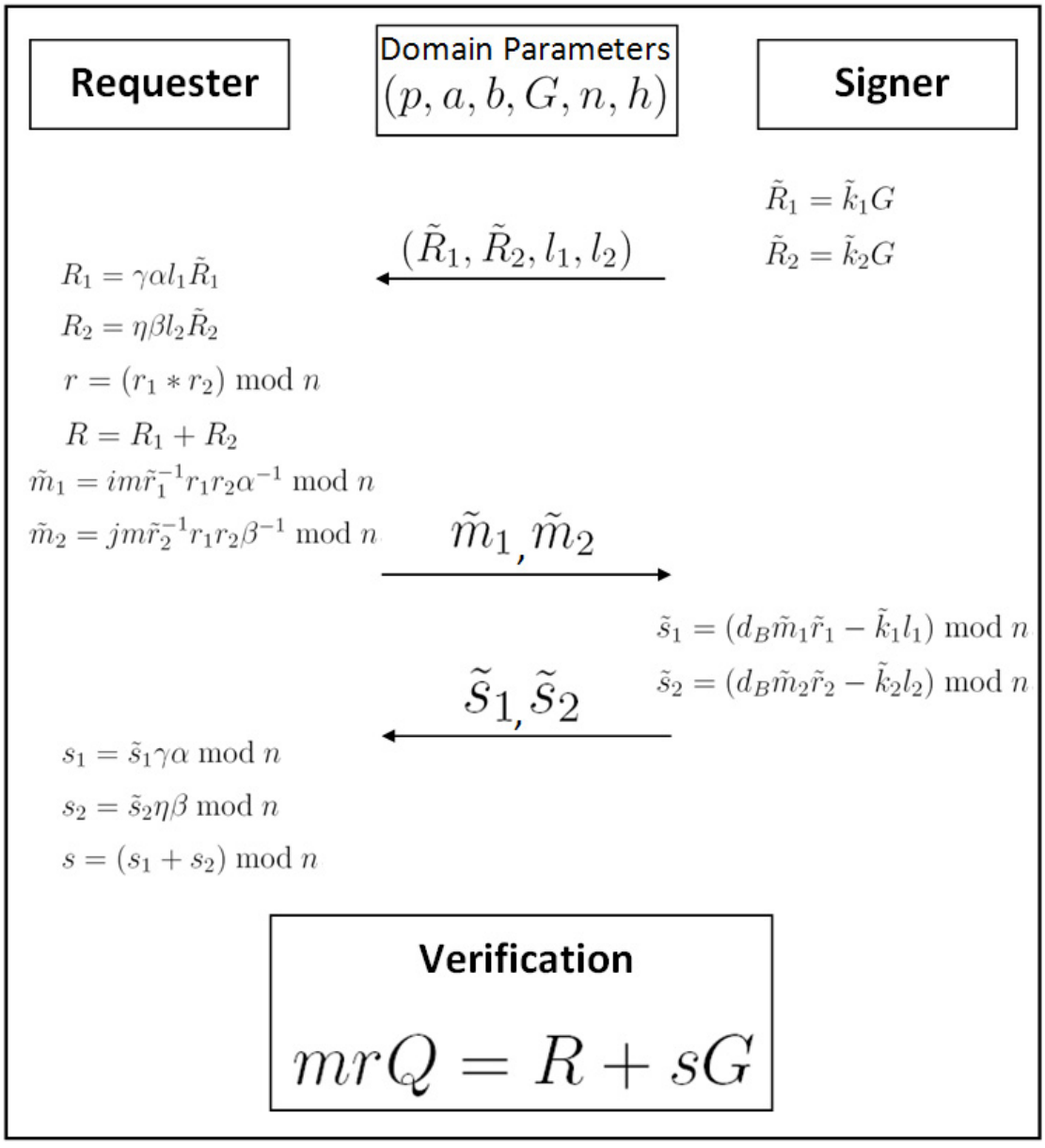


Figure 4.5: Proposed blind signature scheme: BSSDSA3

4.4 Poof of Properties of the Proposed Blind Signature Schemes

In this section, we discuss the *correctness, blindness, untraceability*, and *unforgeability* and some of the important properties of our proposed blind signature schemes.

4.4.1 Correctness

This section proves the correctness of the equations for verifying phases of four proposed BSS.

1. BSSNRS

The verifier has only digital signature (r, s, R) of message m for verification.

$$\begin{aligned} s &= (\tilde{s}\beta + \alpha) \bmod n \\ &= ((\tilde{m}d_B + k)\beta + \alpha) \bmod n \\ &= ((r\beta^{-1}d_B + k)\beta + \alpha) \bmod n \\ &= (rd_B + k\beta + \alpha) \bmod n \end{aligned} \tag{4.42}$$

Now multiplying both sides of (4.42) by generator G, we have

$$\begin{aligned} & sG = rd_BG + k\beta G + \alpha G \\ & \Rightarrow sG = rQ + \tilde{R}\beta + \alpha G \\ & \Rightarrow sG = rQ + R - M \\ & \Rightarrow M = rQ + R - sG \end{aligned}$$

2. BSSDSA1

The correctness of this scheme can be easily verified as follows. The verifier has only digital signature (r, s, R) of message m for verification.

$$\begin{aligned} s &= (\tilde{s}r\tilde{r}^{-1} + \beta m) \bmod n \\ &= ((k\tilde{m} + \tilde{r}d_B)r\tilde{r}^{-1} + \beta m) \bmod n \\ &= (k\tilde{m}r\tilde{r}^{-1} + \tilde{r}d_Br\tilde{r}^{-1} + \beta m) \bmod n \\ &= (k\alpha m\tilde{r}r^{-1}r\tilde{r}^{-1} + d_Br + \beta m) \bmod n \\ &= (k\alpha m + d_Br + \beta m) \bmod n \end{aligned} \tag{4.43}$$

Now multiplying both sides of (4.43) by generator G, we have

$$\begin{aligned}
&sG = k\alpha mG + d_B rG + \beta mG \\
&\Rightarrow sG = \tilde{R}\alpha m + rQ + \beta mG \\
&\Rightarrow sG = m(\tilde{R}\alpha + \beta mG) + rQ \\
&\Rightarrow sG = mR + rQ \\
&\Rightarrow mR = sG - rQ
\end{aligned}$$

3. BSSDSA2

The correctness of this scheme can be easily verified as follows. The verifier has only digital signature (r, s, R) of message m for verification.

$$\begin{aligned}
&s = (s_1 + s_2) \bmod n \\
&= (\tilde{s}_1\tilde{r}_1^{-1}r_1r_2\gamma\alpha + \tilde{s}_2\tilde{r}_2^{-1}r_1r_2\eta\beta) \bmod n \\
&= ((d_B\tilde{m}_1 - \tilde{r}_1k_1l_1)\tilde{r}_1^{-1}r_1r_2\gamma\alpha + (d_B\tilde{m}_2 - \tilde{r}_2k_2l_2)\tilde{r}_2^{-1}r_1r_2\eta\beta) \bmod n \\
&= ((d_Bim\tilde{r}_1r_1^{-1}r_2^{-1}\alpha^{-1} - \tilde{r}_1k_1l_1)\tilde{r}_1^{-1}r_1r_2\gamma\alpha + \\
&\quad (d_Bjm\tilde{r}_2r_1^{-1}r_2^{-1}\beta^{-1} - \tilde{r}_2k_2l_2)\tilde{r}_2^{-1}r_1r_2\eta\beta) \bmod n \\
&= (d_Bim\gamma - k_1l_1r_1r_2\gamma\alpha + d_Bjm\eta - k_2l_2r_1r_2\eta\beta) \bmod n \\
&= (d_Bm(i\gamma + j\eta) - r_1r_2(k_1l_1\gamma\alpha + k_2l_2\eta\beta)) \bmod n \\
&= (d_Bm - r(k_1l_1\gamma\alpha + k_2l_2\eta\beta)) \bmod n
\end{aligned} \tag{4.44}$$

Now multiplying both sides of (4.44) by generator G, we have

$$\begin{aligned}
&sG = (d_Bm - r(k_1l_1\gamma\alpha + k_2l_2\eta\beta))G \\
&\Rightarrow sG = d_BmG - r(k_1l_1\gamma\alpha G + k_2l_2\eta\beta G) \\
&\Rightarrow sG = Qm - r(\tilde{R}_1l_1\gamma\alpha + \tilde{R}_2l_2\eta\beta) \\
&\Rightarrow sG = Qm - r(R_1 + R_2) \\
&\Rightarrow sG = Qm - rR \\
&\Rightarrow rR = sG + Qm
\end{aligned}$$

4. BSSDSA3

The correctness of this scheme can be easily verified as follows. The verifier has only digital signature (r, s, R) of message m for verification.

$$\begin{aligned}
&s = (s_1 + s_2) \bmod n \\
&= (\tilde{s}_1\gamma\alpha + \tilde{s}_2\eta\beta) \bmod n \\
&= ((d_B\tilde{m}_1\tilde{r}_1 - k_1l_1)\gamma\alpha + (d_B\tilde{m}_2\tilde{r}_2 - k_2l_2)\eta\beta) \bmod n \\
&= ((d_Bim\tilde{r}_1^{-1}r_1r_2\alpha^{-1}\tilde{r}_1 - k_1l_1)\gamma\alpha + (d_Bjm\tilde{r}_1^{-1}r_1r_2\beta^{-1}\tilde{r}_2 - k_2l_2)\eta\beta) \bmod n \\
&= ((d_Bimr_1r_2\alpha^{-1} - k_1l_1)\gamma\alpha + (d_Bjmr_1r_2\beta^{-1} - k_2l_2)\eta\beta) \bmod n \\
&= ((d_Bimr\alpha^{-1} - k_1l_1)\gamma\alpha + (d_Bjmr\beta^{-1} - k_2l_2)\eta\beta) \bmod n \\
&= (d_Bimr\alpha^{-1}\gamma\alpha - k_1l_1\gamma\alpha + d_Bjmr\beta^{-1}\eta\beta - k_2l_2\eta\beta) \bmod n \\
&= (d_Bimr\gamma - k_1l_1\gamma\alpha + d_Bjmr\eta - k_2l_2\eta\beta) \bmod n \\
&= (d_Bmr(i\gamma + j\eta) - (k_1l_1\gamma\alpha + k_2l_2\eta\beta)) \bmod n \\
&= (d_Bmr - (k_1l_1\gamma\alpha + k_2l_2\eta\beta)) \bmod n
\end{aligned} \tag{4.45}$$

Now multiplying both sides of (4.45) by generator G, we have

$$\begin{aligned}
&sG = (d_Bmr - (k_1l_1\gamma\alpha + k_2l_2\eta\beta))G \\
&\Rightarrow sG = d_BmrG - (k_1l_1\gamma\alpha + k_2l_2\eta\beta)G \\
&\Rightarrow sG = Qmr - (k_1l_1\gamma\alpha G + k_2l_2\eta\beta G) \\
&\Rightarrow sG = Qmr - (\tilde{R}_1l_1\gamma\alpha + \tilde{R}_2l_2\eta\beta) \\
&\Rightarrow sG = Qmr - (R_1 + R_2) \\
&\Rightarrow sG = Qmr - R \\
&\Rightarrow mrQ = sG + R
\end{aligned}$$

4.4.2 Blindness

This is the one of the important property of the BSS. Because of this property, the Signer signs the document without knowing the contents of the document. In case of David Chaum's scheme, this property is achieved because of the multiplication of the blinding factor r with the message m, i.e., blind message $\tilde{m} = r^e m \bmod n$, where e is the public key of the Signer. Thus, Signer cannot find the original message from the blind message. Similarly, our protocols have achieved this property as follows.

1. BSSNRS

The Requester selects two random numbers α, β and blinds the original message by $\tilde{m} = r\beta^{-1} \bmod n$. The Requester sends the blind message to Signer. As α and β are known to only Requester, hence it is not possible for the Signer to find the message m from the blind message $\tilde{m}$.

2. BSSDSA1

The Requester selects two random numbers α, β and blinds the original message by $\tilde{m} = \alpha m\tilde{r}r^{-1} \bmod n$. The Requester sends the blind message to Signer. As α and β are known to only Requester, hence it is not possible for the Signer to find the message m from the blind message $\tilde{m}$.

3. BSSDSA2

The Requester selects four random numbers $\alpha, \beta, \gamma, \eta$ and blinds the original message by $\tilde{m}_1 = im\tilde{r}_1 r_1^{-1} r_2^{-1}\alpha^{-1} \bmod n$ and $\tilde{m}_2 = jm\tilde{r}_2 r_1^{-1} r_2^{-1}\beta^{-1} \bmod n$. The Requester sends the blind messages $\tilde{m}_1$ and $\tilde{m}_2$ to Signer. As α, β, γ, and η are known to only Requester, hence it is not possible for the Signer to find the message m from the blind messages $\tilde{m}_1$ and $\tilde{m}_2$.

4. BSSDSA3

The Requester selects four random numbers $\alpha, \beta, \gamma, \eta$ and blinds the original message by $\tilde{m}_1 = im\tilde{r}_1^{-1} r_1 r_2\alpha^{-1} \bmod n$ and $\tilde{m}_2 = jm\tilde{r}_2^{-1} r_1 r_2\beta^{-1} \bmod n$. The Requester sends the blind messages age m from the blind messages to Signer. As α, β, γ, and η are known to only Requester, hence it is not possible for the Signer to find the message m from the blind messages $\tilde{m}_1$ and $\tilde{m}_2$.

4.4.3 Untraceability

As per this property, the Signer cannot link a valid signature to the message. The proposed protocols have achieved this property as follows.

1. BSSNRS

The Signer stores the information about all the signatures which are signed by him as $(\tilde{m}_i, \tilde{k}_i, \tilde{R}_i, \tilde{s}_i)$, where $i = 1, ..., n$. When Requester reveals all n valid signatures (m, R, s) to others, the Signer can evaluate $r\beta^{-1}$ ($\tilde{m} = r\beta^{-1} \bmod n$). But, the Signer cannot trace the blind signature to the message without knowing the α and β.

2. BSSDSA1

The Signer stores the information about all the signatures which are signed by him as $(\tilde{m}_i, \tilde{k}_i, \tilde{R}_i, \tilde{s}_i)$, where $i = 1, ..., n$. When Requester reveals all n valid signatures (m, R, s) to others, the Signer can evaluate αr^{-1} ($\tilde{m} = \alpha m\tilde{r}r^{-1} \bmod n$). But, the Signer cannot trace the blind signature to the message without knowing the α and β.

3. BSSDSA2

The Signer stores the information about all the signatures which are signed by him as $(\tilde{m}_i, \tilde{k}_i, \tilde{R}_i, \tilde{s}_i)$, where $i = 1, ..., n$. When Requester reveals all n valid signatures (m, R, s) to others, the Signer can evaluate $i\alpha^{-1}$ ($\tilde{m}_1 = im\tilde{r}_1 r_1^{-1} r_2^{-1}\alpha^{-1} \bmod n$) and $j\beta^{-1}$ ($\tilde{m}_2 = jm\tilde{r}_2 r_1^{-1} r_2^{-1}\beta^{-1} \bmod n$). But the Signer cannot trace the blind signature to the message without knowing the α, β, γ, and η.

4. BSSDSA3

The Signer stores the information about all the signatures which are signed by him as $(\tilde{m}_i, \tilde{k}_i, \tilde{R}_i, \tilde{s}_i)$, where $i = 1, ..., n$. When Requester reveals all n valid signatures (m, R, s) to others, the Signer can evaluate $i\alpha^{-1}$ ($\tilde{m}_1 = im\tilde{r}_1^{-1} r_1 r_2\alpha^{-1} \bmod n$) and $j\beta^{-1}$ ($\tilde{m}_2 = jm\tilde{r}_2^{-1} r_1 r_2\beta^{-1} \bmod n$). But, the Signer cannot trace the blind signature to the message without knowing the α, β, γ, and η.

4.4.4 Unforgeability

BSS are mainly based on IFP, DLP or ECDLP. As per the research, there is no known sub-exponential algorithm which can solve the ECDLP. Hence, the BSS

based on ECDLP are more secure to their counterparts which are developed using IFP or DLP.

1. BSSNRS

As per ECDLP, it is computational infeasible to find the secrete key d_B from $Q = d_B G$. From the verification equation $M = rQ + R - sG$, an attacker can choose randomly another R or s and try to find s or R, but as it is based on the difficulty of ECDLP, hence it is not possible to find another set of R and s form the verification equation as mentioned above.

2. BSSDSA1

As per ECDLP, it is computational infeasible to find the secrete key d_B from $Q = d_B G$. From the verification equation $mR = sG - rQ$, an attacker can choose randomly another R or s and try to find s or R, but as it is based on the difficulty of ECDLP, hence it is not possible to find another set of R and s form the verification equation as mentioned above.

3. BSSDSA2

As per ECDLP, it is computational infeasible to find the secrete key d_B from $Q = d_B G$. From the verification equation $mQ = sG + rR$, an attacker can choose randomly another R or s and try to find s or R, but as it is based on the difficulty of ECDLP, hence it is not possible to find another set of R and s form the verification equation as mentioned above.

4. BSSDSA3

As per ECDLP, it is computational infeasible to find the secrete key d_B from $Q = d_B G$. From the verification equation $mrQ = R + sG$, an attacker can choose randomly another R or s and try to find s or R, but as it is based on the difficulty of ECDLP, hence it is not possible to find another set of R and s form the verification equation as mentioned above.

4.5 Results and Discussion

The proposed schemes have been proved to be correct. They also satisfy the properties of BSS, i.e, blindness, untraceability and unforgeability. All the schemes have been implemented using Java Card Toolkit 2-1-2 and tested in Java Card simulator— *jcwde* (Java Card Workstation Development Environment). The proposed schemes are compared with the known BSS which are given in Table 4.1.

Table 4.1: Comparative statement among blind signature schemes

Scheme	**Basis**	**80-Bit Security Strength**
Chaum [55]	IFP	1024-bit modulus
Cao and Liu [123]	IFP	1024-bit modulus
Camenisch et al. (Version 1) [84]	DLP	$\|p\| = 1024$, $\|q\| = 160$
Camenisch et al. (Version 2) [84]	DLP	$\|p\| = 1024$, $\|q\| = 160$
Proposed schemes	ECDLP	$m = 163$ bits

As per NIST [124], ECDLP-based BSS takes 160 bits for 80-bit security strength, which is comparable to proposed scheme. Lourdes et al. [125] has implemented our proposed scheme and they also found that it takes 163 bits.

With respect to the computational time, the proposed schemes are compared with the known BSS which are given in Table 4.2.

Table 4.2: Time Comparison among blind signature schemes for 80-bit security strength

Scheme	**Time in μs**
Chaum [55]	7058.59
Cao and Liu [123]	8002.65
Camenisch et al. (Version 1) [84]	3352.38
Camenisch et al. (Version 2) [84]	2803.15
Proposed schemes	1356.45

It can be seen from the Table 4.2 that the proposed scheme takes much less computational time than the known BSS. Computational cost of proposed scheme is almost one-third of that of DLP-based BSS and one-sixth of IFP-based BSS.

Chapter 5

Remote User Authentication Protocol using Smart Card

Chapter 5

Remote User Authentication Protocol using Smart Card

The use of information technology and Internet has grown in exponential way. Due to this, service providers share their services through networks. As the resources are not free for all, access control is necessary. So, authentication is the key for information security since, if the authentication mechanism is compromised, the rest of the security measures are bypassed as well. Authentication is the process, in which a user's claim to an identity is checked for accessing the services offered. Conventional user authentication schemes are suited to solve the privacy and security problems for the single client/server architecture environment. Remote user authentication is used to validate the legitimacy of a remote login user.

Password-based remote user authentication schemes are used to check the validity of a login request made by a remote user U to before allowing him/her to access the server resources. In these schemes, the authentication server (AS) and the remote user U share a secret, which is often called as password. With the knowledge of this password, the remote user U uses it to create a valid login request to the AS. AS requires a verification table, which contains the passwords of various users, to check the validity of the login request made by the user U. However, this approach introduces the risk and cost of managing and protecting the password table. If the password verification table is stolen by the adversary, the system will be partially or totally breached.

To overcome this problem, several password authentication schemes with smart card have been proposed. Remote user authentication schemes can be categorized

into hash based, public-key based etc. In this chapter, schemes from each group have been discussed in brief and the proposed scheme is explained.

5.1 Schemes based on Hash Function

Lamport's (LL) [93] and Cheng-Chi Lee et al. (LLH) [99] schemes are explained below.

5.1.1 LL Scheme

In 1981, Lamport introduced the first hash-based password authentication scheme. According to him, an intruder could learn the user secret password and then impersonate him when interacting with the system because of the following reasons

- By gaining access to the information stored inside the system, e.g., reading the system's password file.
- By intercepting the user's communication with the system, e.g., eavesdropping on the line connecting the user's terminal with the system, or observing the execution of the password checking program.
- By the user's inadvertent disclosure of his password, e.g., choosing an easily guessed password.

To overcome these difficulties, he suggested the first user authentication scheme using a verifier table. The scheme is explained below:
He has used a one-way function to encode the password table, by which the first difficulty can be handled. A one-way function is a mapping F from some set of words into itself such that:

1. Given a word x, it is easy to compute $F(x)$.
2. Given a word y, it is not feasible to compute a word x such that $y = F(x)$.

Instead of storing the user's password x, the system stores only the value $y = F(x)$. The user identifies himself by sending x to the system; the system

authenticates his identity by computing $F(x)$ and checking that it equals the stored value y.

To prevent the second weakness, user must use a sequence of passwords x_1, x_2, $..., x_{1000}$, where x_i is the password by which the user identifies himself for the i^{th} time. The system must know the sequence $y_1, y_2, ..., y_{1000}$ where, $y_i = F(x_i)$, and the y_i must be distinct to prevent an intruder from reusing a prior password. There are two obvious schemes for choosing the passwords x_i.

1. All x_i are chosen initially, and the system maintains the entire sequence of values $y_1, y_2, ..., y_{1000}$ in its storage.

2. The user sends the value y_{i+1} to the system during the i^{th} session, after logging on with x_i.

Neither scheme is completely satisfactory, because

1. both the user and the system must store 1000 pieces of information

2. it is not robust— communication failure or interference from an intruder could prevent the system from learning the correct value of y_{i+1}.

Lamport presents a method that combines the best features of both schemes without these drawbacks. As per him, let the i_{th} password $x_i = F^{1000-i}(x)$ for some fixed word x, where F^n denotes n successive applications of F. Thus, the sequence of 1000 passwords is

$$F^{999}(x), ..., F(F(F(x))), F(F(x)), F(x), x$$

The sequence of y_i needed by the system to authenticate these passwords is

$$F^{1000}(x), ..., F(F(F(x))), F(F(x)), F(x)$$

Since it is feasible to compute F^n for $n < 1000$, property of the one-way function implies that these y_i are distinct. It follows from definition of the x_i that $y_i = x_{i-1}$ for $i > 1$. In other words, each user password is the value needed by the system to authenticate the next password. Hence, the system must initially be given the value $y_1 = F^{1000}(x)$ and need subsequently remember only the last password sent by the user.

5.1.2 LLH Scheme

The scheme is based on the hash functions. There are three phases in the scheme—registration, user authentication, and change password phases.

The Registration Phase

In the registration phase, each user U_i must randomly choose identity ID_i and password PW_i, and then calculate a password digest $HPW_i = H(ID_i, PW_i)$, where $H()$ is a collision resistant hash function, such as SHA-1. U_i sends ID_i and HPW_i to a server. The server stores ID_i and HPW_i in verification table.

The Authentication Phase

1. The user U_i enters his/her identity ID_i and password PW_i to the client. The client calculates $HPW_i = H(ID_i, PW_i)$, where $H()$ is a hash function with output message in 512 bits length. The client randomly chooses a number R_c with 512 bits length and sends ID_i and $R_c \oplus HPW_i$ to the server.

2. After receiving ID_i and $R_c \oplus HPW_i$, the server retrieves the user's password digest HPW_i from the verification table and then obtains R_c by computing $(R_c \oplus HPW_i) \oplus HPW_i$. Next, the server randomly generates a number R_s with 512 bits length and calculates $R_s \oplus HPW_i$ and $AUTH' = H(HPW_i, R_c, R_s)$. The server sends $R_s \oplus HPW_i$ to the client.

3. After receiving $R_s \oplus HPW_i$, the client retrieves R_s by computing $(R_s \oplus HPW_i) \oplus HPW_i$. Next, the client calculates

$$AUTH = H(HPW_i, R_c, R_s) \tag{5.1}$$

   The client sends ID_i and $AUTH$ to the server.

4. After receiving ID_i and $AUTH$, the server compares $AUTH$ with $AUTH'$. If it is equal, the server accepts the accessing resources request. Otherwise, it rejects the client's request.

The Change Password Phase

Steps 1 and 2 are the same as that of the user authentication phase. It is assumed that U_i wants to change his/her password PW_i to $NewPW_i$.

3. After receiving $R_s \oplus HPW_i$, the client retrieves R_s by computing $(R_s \oplus HPW_i) \oplus HPW_i$. Next, the client calculates

$$NewHPW_i = H(ID_i, NewPW_i) \quad (5.2)$$

$$AUTH = H(HPW_i, R_c, R_s) \quad (5.3)$$

$$Mask = NewHPW_i \oplus H(HPW_i, R_c + 1, R_s) \quad (5.4)$$

   The client sends $ID_i, AUTH$, and $Mask$ to the server.

4. After receiving $ID_i, AUTH$, and $Mask$, the server retrieves the user's HPW_i from the verification table, and then compares $AUTH$ with $AUTH'$. If it is equal, the server accepts the client to change the user's password and obtains new password digest $NewHPW_i$ as follows:

$$NewHPW_i = Mask \oplus H(HPW_i, R_c + 1, R_s) \quad (5.5)$$

   Then the server replaces the old HPW_i with the new password digest $NewHPW_i$ in verification table.

The above schemes are hash-based remote user authentication. These schemes take low computation cost and are computationally viable for implementation in a handheld device like smart card. However, the schemes primarily suffer from password guessing, stolen-verifier and denial-of-service attacks. For this, public-key based authentication schemes are proposed which meet higher security requirements. We will discuss two scheme based on public-key in brief.

5.2 Schemes based on Public Key Cryptosystem

Wen-Her Yang et al. (YS) [102] and Hwang et. al (ML) [105] schemes are explained below.

5.2.1 YS Scheme

In this scheme, Wen-Her Yang et al. assume that there exists a trusted key information center in the network to issue personalized smart cards to users when joining the system. Their scheme is divided into three phases. In the registration phase, the key information center sets up the authentication system and issues smart cards to the users who requests registration. In the login phase, a user attaches his smart card to a terminal and keys in his identifier ID and password PW. Then the terminal sends a login request message to the remote host. In the verification phase, the remote host verifies the correctness of submitted message and determines whether the login request should be accepted or not.

Registration Phase

The key information center is not responsible for authenticating users, but for generating key information, issuing smart cards to new users and serving password-changing request for registered users. Let U_i denote the i^{th} user who submits his identifier ID_i; and chosen password PW_i, to the key information center to request for registration. Here, PW_i must be sent over a secure channel. Upon receipt of the request, the key information center will perform the following steps:

1. Generate two large prime numbers p and q, and let $n = pq$. For security reasons, the length of p and q is recommended to be 512 bits at least.

2. Choose a prime number e and an integer d which satisfy

$$e.d(\mathrm{mod}(p-1).(q-1)) = 1. \tag{5.6}$$

   Here, e is the public key of the key information center that should be published, and d is the secret key that must be kept privately.

3. Find an integer g which is a primitive element in both $GF(p)$ and $GF(q)$, where g is the system's public information.

4. Calculate the user's secret information S_i as

$$S_i \equiv ID_i^d \bmod n. \tag{5.7}$$

According to the RSA algorithm , the following equation would be obtained.

$$ID_i \equiv S_i^e \bmod n \tag{5.8}$$

Even if one knows ID_i, e, and n, it is hard to crack S_i without the knowledge of d. This is a discrete logarithm problem. The integer d can be evaluated only when n is factorized to p and q, which is very difficult because the length of n is 1024 bits.

5. Generate the smart card's identifier CID_i, of U_i and compute h_i by

$$h_i = g^{PW_i d} \bmod n \tag{5.9}$$

Here CID_i is used for validating the legality of smart cards in the verification phase.

6. Write $n, e, g, ID_i, CID_i, S_i$, and h_i to the memory of smart card and issue the card to U_i.

Once the authentication system is set up, the key information center is not needed except when new users request to join, or registered users request to change passwords. When a new user requests to join, the center repeats step 4 through 6.

Login Phase

When U_i wishes to login a remote host, he must insert the smart card into a card reader and enter his identity ID_i' and password PW_i'. If ID_i' is identical to the ID_i which is kept in the memory of the smart card, then smart card performs the following steps:

1. Generate a random number r_i and calculate the following two integers:

$$X_i = g^{r_i PW_i} \bmod n$$
$$Y_i = S_i h_i^{r_i \cdot f(CID_i, T)} \bmod n$$

where T is the current time used as a time-stamp and $f(x, y)$ is a one-way function. The one-way function is a function relatively easy to compute but significantly harder to undo or reverse.

2. Send a login request message M containing $ID_i, CID_i, X_i, Y_i, n, e, g$ and T to the remote host.

Verification Phase

The verification phase is executed by the remote host to determine whether U_i is allowed to login or not. Let T' be the time when the remote host receives the message M. Upon receipt of message M, the remote host will perform the following steps to verify the correctness of M.

1. Verify that ID_i is a valid user identity and CID_i is a legal smart card identity. If not, the login request is rejected.

2. Compare T with T'. If the difference between T and T' is longer than the valid period, M is considered as an invalid message and the host computer will reject the login request. According to different network environments, the length of the valid period can be adjusted.

3. It checks whether the following equation holds:

$$Y_i^e = ID_i X_i^{f(CID_I,T)}$$

The equation will hold when the password PW_i', keyed in by U_i, matches PW_i registered in the key information center. That is because:

$$\begin{aligned} Y_i^e &= (S_i h_i^{r_i f(CID_i,T)})^e \\ &= S_i^e (h_i^{r_i f(CID_i,T)})^e \\ &= S_i^e (g^{PW_i d})^{r_i f(CID_i,T)e} \\ &= ID_i (g^{ed})^{r_i PW_i f(CID_i,T)} \\ &= ID_i g^{r_i PW_i f(CID_i,T)} \end{aligned}$$

Therefore,

$$ID_i X_i^{f(CID_i,T)} = ID_i g^{r_i PW_i f(CID_i,T)}$$

4. If the equation holds, the remote host believes that the message M is sent by U_i, and the password PW_i' matches PW_i. Therefore, U_i, is allowed to login the remote host, otherwise the login request is rejected.

5.2.2 HL Scheme

The user authentication scheme is based on ElGamal's public key cryptosystem which uses smart card. The scheme is divided into three phases— registration,

login and authentication phase. In the registration phase, the user U sends a request to the AS for registration. The AS issues a smart card and a password to every legal user through a secure channel. In the login phase, when the user U wants to access the AS, she/he inserts her/his smart card to the smart card reader and then keys the identity and the password to access services. In the authentication phase, the AS checks the validity of the login request.

Registration Phase

A user U submits her/his ID to the AS. AS computes the password PW for the user U as

$$PW = ID^{x_s} \bmod p$$

where, x_s is a secret key maintained by the AS and p is a large prime number. AS provides a password PW and a smart card to the user U through a secure channel. The smart card contains the public parameters (f, p), where f is a one-way function.

Login Phase

User U attaches her/his smart card to the smart card reader and keys ID and PW. The smart card performs the following operation:

1. Generate a random number r.
2. Compute $C_1 = ID^r \bmod p$
3. Compute $t = f(T \oplus PW) \bmod p - 1$, where T is the current date and time of the smart card reader.
4. Compute $M = ID^t \bmod p$.
5. Compute $C_2 = M(PW)^r \bmod p$.
6. Sends a login request $C = (ID, C_1, C_2, T)$ to AS.

Authentication Phase

Assume that AS receives the message C at time T_c, where T_c is the current date and time at AS. Then the AS takes the following action:

1. Check the format of ID. If the identity format is not correct, then AS will reject this login request.

2. Check, whether $T_c - T \leq \Delta T$, where ΔT is the legal time interval due to transmission delay; if not, then rejects the login request C.

3. Check, if $C_2(C_1^{x_s})^{-1} = (ID)^{f(T \oplus PW)} \bmod p$, then the AS accepts the login request; otherwise, the login request will be rejected.

5.3 Proposed Protocols

EC cryptosystems gives more security with less bit size key and computationally faster than other cryptosystems. Because of these reasons, remote user authentication schemes have been proposed which are based on ECDLP and thus, can easily be implemented in smart card. Before describing the schemes, we will discuss the ECC based on ElGamal, then we will discuss the proposed schemes. The first scheme is based on HL scheme. The scheme is divided into three phases— registration, login and authentication phase. Beside these phases, password change phase is used to enable the user to change his password as per his requirement with out help of remote system.

Elliptic Curve Cryptosystem based on Elgamal

Suppose Alice wishes to send a message M to Bob. First, she imbeds the value M onto the elliptic curve E , i.e., she represents the plaintext M as a point $P_m \in E$. Now she must encrypt P_m. Let d_B denote Bob's secret key. Alice first chooses a random integer k and sends Bob a pair of points (C_1, C_2) on E:

$$C_1 = kG$$
$$C_2 = P_m + kQ$$

To decrypt the cipher text, Bob computes

$$\begin{aligned} & C_2 - d_B C_1 \\ \Rightarrow & P_m + kQ - d_B(kG) \\ \Rightarrow & P_m + kQ - k(d_B G) \\ \Rightarrow & P_m + kQ - kQ \\ \Rightarrow & P_m \end{aligned}$$

5.3.1 User Authentication Protocol 1 (UAP1)

The notations used in UAP1 are given below:

U	Remote user
ID	Identity of the remote user
PW	Password corresponding to the registered identity
AS	Authentication Server
$f()$	Cryptographic one-way hash function

Registration Phase

Initially, the curve domain parameters (p, a, b, G, n, h) as mentioned in Chapter 3 must be agreed upon by both the U and the AS. The user U converts his identity and chosen password to point in Elliptic curve as ID and submits to the system. This secret information must be sent over a secure channel. Upon receiving the registration request, AS calculates the password PW as follows.

$$PW = d_s ID$$

where, d_s is a secret key maintained by AS.

The registration centre issues a smart card which contains the public parameter (f, n, G, Q), where f is a one-way function and Q is the public key of AS, i.e., $Q = d_s G$. The registration centre also delivers PW to the user through a secure channel. The smart cards possessed by all users will contain the same data and functions, i.e., (f, n, G, Q).

Login Phase

Upon login, U attaches smart card to his/her input device. Then he enters her/his ID and PW to the device. The smart card will perform the following operations:

1. Select r randomly between $[1, n-1]$.
2. Compute $C_1 = rID$.
3. Compute $t = f(T \oplus PW) \bmod n$, where T is the current date and time of the input device.
4. Compute $M = tID$.
5. Compute $C_2 = M + rPW$.
6. Send a message C consists of (ID, C_1, C_2, T) to AS.

Authentication Phase

Upon receive of message C, AS authenticates the login user as follows :

1. Let AS receives the message C sent from U at T' , where $\tilde{T}$ is the current date and time of the system.
2. Test the validity of ID. If the format of the ID is incorrect, then the AS rejects the login user.
3. Test, whether $T' - T \leq \Delta T$, where ΔT is the legal time interval due to transmission delay, if not, then rejects the login request C.
4. Check, if $C_2 - d_s C_1 = M$, then AS accepts otherwise rejects the login user.

$$\begin{aligned} & C_2 - d_s C_1 \\ & = M + rPW - d_s C_1 \\ & = M + rPW - d_s rID \\ & = M + rd_s ID - d_s rID \\ & = M \end{aligned}$$

The scheme is described in the Figure 5.1.

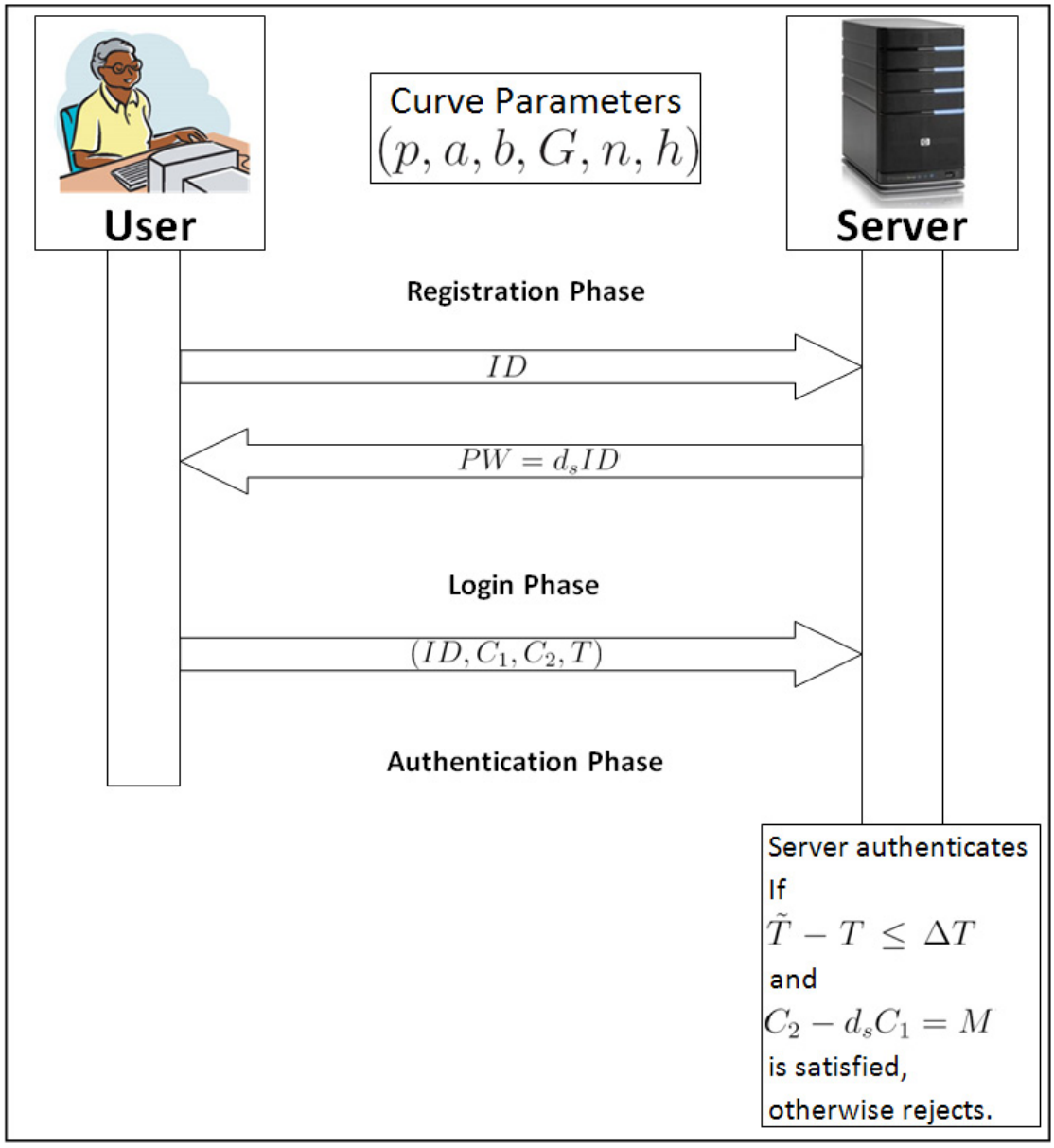


Figure 5.1: User Authentication Protocol 1

Security analysis

As the proposed scheme is based on ECDLP, so it is not possible for attacker to find the secret key d_s of AS from PW where $PW = d_s ID$. It is also difficult for the attacker to find the randomly selected r from $C_1 = rID$ in the login phase. For the attacker to pass through the step 2 of the authentication phase, he must change T' into T'' such that $(T'' - T') \leq \Delta T$. Once T is changed, the step 3 in the authentication phase is failure unless either t or C_2 has been changed accordingly.

Therefore, the proposed scheme is secure to withstand the replay attack.

5.3.2 User Authentication Protocol 2 (UAP2)

The notations used in UAP2 are given below:

U	Remote user
ID	Identity of the remote user
PW	Password corresponding to the registered identity
AS	Authentication Server
$f()$	Cryptographic one-way hash function
SID	Shadow Identity
IS	Identity string that includes name, unique number etc.

Registration Phase

Initially, the curve domain parameters (p, a, b, G, n, h) as mentioned in Chapter 3 must be agreed upon by both the U and AS. The scheme employs the concept of hiding identity to prevent from masquerading attack. Assume that this phase is executed over a secure channel. First, U submits her/his identity string IS, to the AS for registration, where IS, is U's unique identity string that includes name, unique number etc. The remote server computes (SID, PW) for the registering user after her/his identity IS, is identified as

$$SID = Sed(IS)$$
$$PW = d_s SID$$

where, $Sed()$ is a shadowed identity of the device which only is possessed with the remote server; SID is U's shadowed identity which can be disclosed. Furthermore, the remote server issues the smart card and (SID, PW) to U. The registration centre issues a smart card which contains the public parameter (f, n, G, Q), where f is a one-way function and Q is the public key of AS, i.e., $Q = d_s G$. The registration centre also delivers PW to the user through a secure channel. The smart cards possessed by all users will contain the same data and functions, i.e., (f, n, G, Q).

Login Phase

Upon login, U attaches smart card to his/her input device. Then he enters her/his SID and PW to the device. The smart card will perform the following operations:

1. Select r randomly between $[1, n-1]$.
2. Compute $C_1 = rSID$.
3. Compute $t = f(T \oplus PW) \bmod n$, where T is the current date and time of the input device.
4. Compute $M = tID$.
5. Compute $C_2 = M + rPW$.

Send a message C consists of (SID, C_1, C_2, T) to AS.

Authentication Phase

Upon receive of message C, AS authenticates the login user as follows:

1. Let AS receive the message C sent from U at $\tilde{T}$, where $\tilde{T}$ is the current date and time of the system.
2. Test the validity of ID . If the format of the ID is incorrect, then the AS reject the login user.
3. Test, whether $\tilde{T} - T \leq \Delta T$, where ΔT is the legal time interval due to transmission delay, if not, then rejects the login request C.
4. If $C_2 - d_s C_1 = M$, then the AS accept otherwise reject the login user.

$$
\begin{aligned}
& C_2 - d_s C_1 \\
\Rightarrow\ & M + rPW - d_s C_1 \\
\Rightarrow\ & M + rPW - d_s rSID \\
\Rightarrow\ & M + rd_s SID - d_s rSID \\
\Rightarrow\ & M
\end{aligned}
$$

The scheme is described in Figure 5.2.

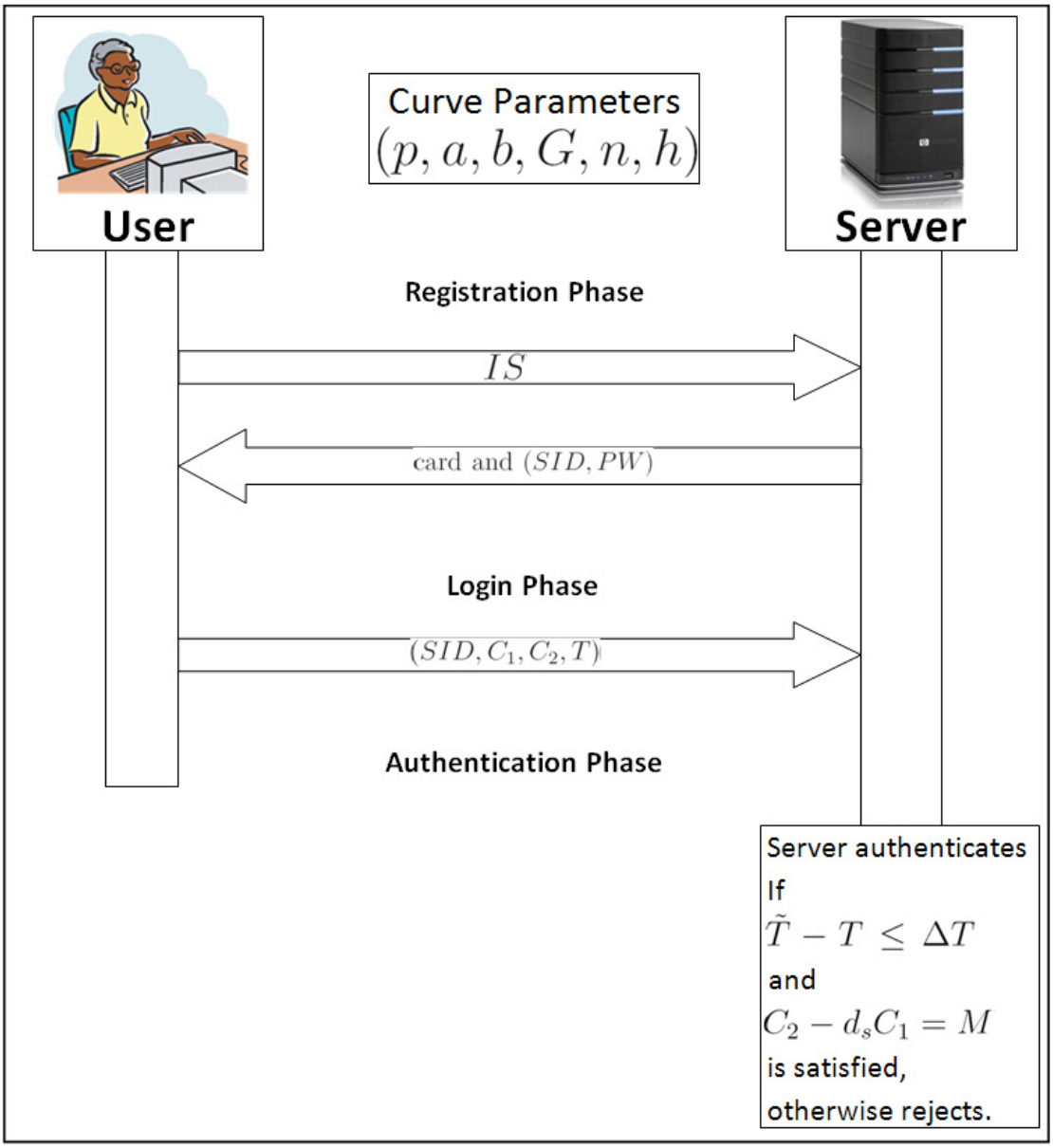


Figure 5.2: User Authentication Protocol 2

Security analysis

Assume that an evil user U_m can intercepts C consists of (SID, C_1, C_2, T) from a public network. Now, U_m submits her/his $IS = kIS$ to AS to register for masquerading as U_l. Upon receiving the registration message IS_l, from U_m, the remote server will reject the registration request, because the format of IS_l, is incorrect which must include name, unique number etc. for identifying IS_m, which is maintained by the user U_m, and AS secretly. Thus, U_l cannot masquerade as

U_m to login and access the remote server.

5.4 Result and Discussion

The correctness of the proposed algorithms are verified and security analysis has been done in previous section. It has been shown that the proposed schemes withstand masquerade and replay attack. The proposed schemes are based on ECDLP. Hence, they achieve the same security with fewer bits key as compared to IFP- and DLP-based scheme. In addition, they have low computation requirements. Thus, these are more suitable in the smart card based application. As we are using smart card for authentication, verifier table is not required. Hence, it reduces the maintenance cost of a verifier table.

Chapter 6

Conclusion and Future Work

Chapter 6

Conclusion and Future Work

Protocols like Key Management, Large Message Encryption, Digital and Blind Signature, Offline Signature and User Authentication Protocols are quite common in ISS. For the last half a century, researchers across the globe have been working to protect the information from attackers and quite significant volumes of literature are available in this regard. Some of the works are based on IFP and DLP, while very little work has been done using ECDLP in case of Large Message Encryption, Digital and Blind Signature and User Authentication Protocols. With regards to Key Management lots of work have been done, but AS behaves as monitoring server. In case of offline signature verification scheme, extensive work have been done to detect random and simple forgeries, but very few research has been done to verify the skilled forgery. Owing to this, these problems are still open and needs substantial research.

In this thesis, attempts have been made to design protocols for ISS. Chapter 2 proposes a three party protocol which is secure against online and dictionary attacks. It provides host and server authentication. As a result, man-in-the-middle attack is averted. It also withstands malicious insider attack. Hosts are not forced to store plaintext version of password at server. Due to this prevention mechanism, online and offline guessing attacks are defeated. Proposed protocol does not make use of any public key infrastructure. Thus, it reduces the implementation cost.

ECDLP is used in Chapter 3 to encrypt large message. It has been shown that the proposed protocols are computationally faster than the conventional El-Gamal scheme. Thus these schemes can be used to encrypt the long message as

compared to conventional encryption systems. As ECDLP is used for encryption, hence the protocols require less computational power, memory and communication bandwidth giving it clear edge over the traditional crypto-algorithm. It has been proved that the proposed protocols are secure against the chosen-plain text attack. It is also shown that the proposed protocols take considerably less computation time than the ElGamal-like ECDLP cryptosystem.

Four proposed BSS based on ECDLP are presented in Chapter 4. These schemes have shown that the properties of BSS which include correctness, blindness, unforgebility and untraceable are achieved. As the schemes are based on ECDLP, they achieve the same security with fewer bits key as compared to their counterpart like IFP and DLP based schemes. In addition, they also require low-computation time. The proposed schemes have been implemented and compared with known BSS and it is found that the proposed schemes outperform others.

Two remote user authentication protocols—UAP1 and UAP2 using smart card are presented in Chapter 5. As both the schemes are based on ECDLP, they achieve the same security level with fewer bits key as compared to RSA and ElGamal. The proposed schemes do not require verifier table and allows the user to choose their passwords. As the schemes are based on ECDLP, they require less computation than their counterpart based on IFP or DLP. Because of these properties, they can be easily implemented in applications where smart cards are being used. They also withstand message replaying attack.

Scope for Further Research

The research findings made out of this thesis has opened several auxiliary research directions, which can be further investigated. The proposed key management scheme, which is used to share secret key between the sender and receiver and can be extended to share secret key between group of users. Large message encryption protocol and BSS may be improved to be used in wireless sensor and adhoc networks. Further scope is there to implement user authentication protocol using hyper-elliptic cryptosystem.

Bibliography

Bibliography

[1] A Certicom Whitepaper. Remarks on the security of the elliptic curve cryptosystem, July 2000.

[2] William Stallings. *Cryptography and Network Security: Principles and Practice*. Pearson Education, 2002.

[3] J. Zadrony, J. Kacprzyk, and K. Floisand. Internet and www - new opportunities for information technology and soft computing. In *Proceedings of IFSA'97 - Seventh International Fuzzy Systems Association World Congress*, volume 4, pages 316–319, 1997.

[4] K. Swigger, F. Alpaslan, R. Brazile, B. Harrington, and Xiaobo Peng. The challenges of international computer-supported collaboration. In *34th Annual Frontiers in Education, FIE 2004*, volume 3, pages 13–18, Oct. 2004.

[5] Rajwinder Singh and A. K. Sarje. Security scheme for mobile agent system in e- commerce scenario, 2008. `http://icsa.cs.up.ac.za/issa/2005/Proceedings/Research/067\_Article.pdf`.

[6] Behrouz A. Forouzan. *Cryptography & Network Security*. McGraw-Hill, Inc., New York, NY, USA, 2008.

[7] G.P. Biswas. An efficient technique of signature computation and the minimization of its aliasing. In *Proceedings of National Seminar on Advances in Mathematical, Statistical and Computational methods in science and Technology*, 2001.

[8] Yao Chen, Shantanu Das, Pulak Dhar, Abdulmotaleb El-Saddik, and Amiya Nayak. Detecting and preventing ip-spoofed distributed dos attacks. *International Journal of Network Security*, 7(1):69–80, 2008. `http://ijns.femto.com.tw/contents/ijns-v7-n1/ijns-2008-v7-n1-p69-80.pdf`.

[9] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22:644–654, Nov 1976.

[10] Jean francois Raymond and Anton Stiglic. Security issues in the diffie-hellman key agreement protocol, 2000. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.639`.

[11] Tianpu Jiang, Yongmin Hou, and Shibao Zheng. Secure communication between set-top box and smart card in dtv broadcasting. *IEEE Transactions on Consumer Electronics*, 50(3):882–886, aug. 2004.

[12] Eun-Jun Yoon and Kee-Young Yoo. A new secure key exchange protocol between stb and smart card in dtv broadcasting. In *WISI*, pages 165–166, 2006.

[13] Song-Hee Lee, Nam-Sup Park, Soo-Kyun Kim, and Jin-Young Choi. Cryptanalysis of secure key exchange protocol between stb and smart card in iptv broadcasting. In *ISA '09: Proceedings of the 3rd International Conference and Workshops on Advances in Information Security and Assurance*, pages 797–803, Seoul, Korea, 2009. Springer-Verlag, Berlin, Heidelberg.

[14] Lui Xiumei, Zhou Fu-cai, and Chang Gui-ran. A verifier-based key exchange protocol in cross-realm setting. In *NSWCTC '09: Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, pages 350–353, Washington, DC, USA, 2009. IEEE.

[15] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password based protocols secure against dictionary attacks. In *IEEE Symposium on Research in Security and Privacy*, pages 72–84, 1992.

[16] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, pages 244–250, New York, NY, USA, 1993. ACM.

[17] David P. Jablon. Strong password-only authenticated key exchange. *SIGCOMM Comput. Commun. Rev.*, 26(5):5–26, 1996.

[18] D.P. Jablon. Extended password key exchange protocols immune to dictionary attack. In *Proceedings Sixth IEEE workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 248–255, jun 1997.

[19] Stefan Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Proc. of the Security Protocols Workshop, LNCS 1361*, pages 79–90. Springer-Verlag, 1997.

[20] Yun Ding and Patrick Horster. Undetectable on-line password guessing attacks. *SIGOPS Oper. Syst. Rev.*, 29(4):77–86, 1995.

[21] Li Gong. Optimal authentification protocols resistant to password guessing attacks. In *Proceedings of the Eighth IEEE Computer Security Foundations Workshop 1995*, pages 24–29, jun 1995.

[22] S. Keung and Kai-Yeung Siu. Efficient protocols secure against guessing and replay attacks. In *International Conference on Computer Communications and Networks*, page 105, Los Alamitos, CA, USA, 1995. IEEE Computer Society.

[23] Michael Steiner, Gene Tsudik, and Michael Waidner. Refinement and extension of encrypted key exchange. *SIGOPS Oper. Syst. Rev.*, 29(3):22–30, 1995.

[24] Taekyoung Kwon, Myeongho Kang, and Jooseok Song. An adaptable and reliable authentication protocol for communication networks. In *Proceedings of Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '97*, volume 2, pages 737–744, apr 1997.

[25] Chun-Li Lin, Hung-Min Sun, and Tzonelih Hwang. Three-party encrypted key exchange: attacks and a solution. *SIGOPS Oper. Syst. Rev.*, 34(4):12–20, 2000.

[26] Chun-Li Lin, Hung-Min Sun, M. Steiner, and T. Hwang. Three-party encrypted key exchange without server public-keys. *IEEE Communications Letters*, 5(12):497–499, Dec 2001.

[27] Maurizio Kliban Boyarsky. Public-key cryptography and password protocols: the multi-user case. In *CCS '99: Proceedings of the 6th ACM conference on Computer and communications security*, pages 63–72, New York, NY, USA, 1999. ACM.

[28] Shai Halevi and Hugo Krawczyk. Public-key cryptography and password protocols. *ACM Trans. Inf. Syst. Secur.*, 2(3):230–268, 1999.

[29] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In *Proceedings of the 6th IMA International Conference on Cryptography and Coding*, pages 30–45, London, UK, 1997. Springer-Verlag.

[30] Her-Tyan Yeh and Hung-Min Sun. Simple authenticated key agreement protocol resistant to password guessing attacks. *ACM SIGOPS Operating System Review*, 36(4):14–22, 2002.

[31] Gene Tsudik and Els Van Herreweghen. On simple and secure key distribution. In *Proceedings of 1993 ACM Conference on Computer and Communications Security*, pages 49–57. ACM Press, 1993.

[32] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-bit block cipher. In *First Advanced Encryption Standard (AES) Conference*, 1998.

[33] Eli Biham, Orr Dunkelman, and Nathan Keller. Differential-linear cryptanalysis of serpent. In *FSE*, pages 9–21, 2003.

[34] FIPS. Data encryption standard. Federal Information Processing Standards, 1977. National Bureau of Standard.

[35] M. Smid and D. Branstad. The data encryption standard: Past and future. *Proceedings of the IEEE*, 76(5):550–559, May 1988.

[36] FIPS. Advanced encryption standard. Federal Information Processing Standards, Nov 2001. Publication 197.

[37] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[38] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb 1978.

[39] Chi-Sung Laih and Sung-Ming Yen. Improved digital signature algorithm. *IEEE Trans. Comput.*, 44(5):729–730, 1995.

[40] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[41] H. Ong, Claus-Peter Schnorr, and Adi Shamir. An efficient signature scheme based on quadratic equations. In *STOC*, pages 208–216, 1984.

[42] Kaisa Nyberg and Rainer A. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. *Des. Codes Cryptography*, 7(1-2):61–81, 1996.

[43] Qin Yanlin and Wu Xiaoping. New digital signature scheme based on both ecdlp and ifp. In *International Conference on Computer Science and Information Technology*, pages 348–351, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[44] Daniel R. Brown. Generic groups, collision resistance, and ecdsa. *Des. Codes Cryptography*, 35(1):119–152, 2005.

[45] Jun Yang and Xianze Yang. A new variant of the diffie-hellman key-exchange protocol based on block triangular matrix groups. In *Intelligent Information Hiding and Multimedia Signal Processing, International Conference on*, volume 0, pages 1277–1281, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[46] Kyung-Sang Sung, Hoon Ko, and Hae-Seok Oh. Xml document encrypt implementation using elliptic curve cryptosystem. In *Convergence Information Technology, International Conference on*, volume 0, pages 2473–2478, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[47] Jorge Guajardo, Rainer Blümel, Uwe Krieger, and Christof Paar. *Public Key Cryptography, LNCS*, volume 1992, chapter Efficient implementation of elliptic curve cryptosystems on the TI MSP430x33x family of microcontrollers, pages 365–382. Springer Berlin / Heidelberg, 2001.

[48] Ching-Nung Yang, Chung-Chun Wang, and Tse-Shih Chen. Visual Cryptography Schemes with Reversing. *The Computer Journal*, 51(6):710–722, 2008. `http://comjnl.oxfordjournals.org/cgi/reprint/51/6/710.pdf`.

[49] M. Prashant, R. Siddharth, and R. Kumar. Formulation of an encryption algorithm on the basis of molecular genetics and image patterns. In *Proceedings of Third International Conference on Computational Intelligence and Multimedia Applications, 1999. ICCIMA '99*, pages 76–80, 1999.

[50] Vijay Laxmi, Mudassar N. Khan, Sarath S. Kumar, and Manoj Singh Gaur. Buyer seller watermarking protocol for digital rights management. In *SIN '09: Proceedings of the 2nd international conference on Security of information and networks*, pages 298–301, New York, NY, USA, 2009. ACM.

[51] D. S. Adane and S. R. Sathe. Secured mobile agent communication. *International Journal of Engineering Research & Industrial Applications (IJERIA)*, 1(5):14–22, 2008.

[52] Min-Shiang Hwang, Chin-Chen Chang, and Kuo-Feng Hwang. An elgamal-like cryptosystem for enciphering large messages. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):445–446, Mar/Apr 2002.

[53] Yuh-Dauh Lyuu and Ming-Luen Wu. An elgamal-like cryptosystem for enciphering large messages. *WSEAS Transactions on Information Science and Applications*, 1(4):1079–1081, Oct 2004.

[54] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[55] David Chaum. Blind signatures for untraceable payments. In *CRYPTO'82*, pages 199–203, 1982.

[56] Chun-I Fan, Wei-Kuei Chen, and Yi-Shung Yeh. Randomization enhanced chaum's blind signature scheme. *Computer Communications*, 23(17):1677–1680, 2000. `http://www.sciencedirect.com/science/article/B6TYP-41ST54T-D/2/ac77b91ac4e0b895cdc36b3ae63eb0ba`.

[57] W. S. Juang and C. L. Lei. Partially blind threshold signatures based on discrete logarithm. *Computer Communications*, 22:73–86, 1999.

[58] Zuhua Shao. Improved user efficient blind signatures. *Electronics Letters*, 36(16):1372–1374, aug 2000.

[59] Ling Zhang, Jian ping Yin, and Yu bin Zhan. An anonymous digital cash and fair payment protocol utilizing smart card in mobile environments. In *Fifth International Conference on Grid and Cooperative Computing Workshops, 2006 (GCCW'06)*, pages 335–340, Oct 2006.

[60] J. Jen-Rong Chen, An-Pin Chen, and R. Wen-Mao Lin. A novel blind signature scheme possessed with dual protections. In *IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, 2003. Proceedings*, pages 123–127, Oct 2003.

[61] C. Popescu and H. Oros. An off-line electronic cash system based on bilinear pairings. In *14th International Workshop on Systems, Signals and Image Processing, 2007 and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services*, pages 438–440, June 2007.

[62] Wang Zhan-gang and Wan Zhen-kai. A secure off-line electronic cash scheme based on ecdlp. In *First International Workshop on Education Technology and Computer Science, 2009(ETCS'09)*, volume 2, pages 30–33, March 2009.

[63] S. Ibrahim, M. Kamat, M. Salleh, and S.R.A. Aziz. Secure e-voting with blind signature. In *4th National Conference on Telecommunication Technology, 2003 (NCTT2003)*, pages 193–197, Jan 2003.

[64] Sung-Hyun Yun and Sung-Jin Lee. An electronic voting scheme based on undeniable blind signature scheme. In *IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, 2003*, pages 163–167, Oct 2003.

[65] Seo-Il Kang and Im-Yeong Lee. A study on the electronic voting system using blind signature for anonymity. In *International Conference on Hybrid*

Information Technology, 2006 (ICHIT'06), volume 2, pages 660–663, Nov 2006.

[66] Lina Wang, Jingli Guo, and Min Luo. A more effective voting scheme based on blind signature. In *International Conference on Computational Intelligence and Security, 2006*, volume 2, pages 1507–1510, Nov 2006.

[67] Behnam Kharchineh and Mehdi Ettelaee. A new electronic voting protocol using a new blind signature scheme. In *Second International Conference on Future Networks, 2010 (ICFN'10)*, pages 190–194, Jan 2010.

[68] David Chaum. Blind signature system. In *CRYPTO*, page 153, 1983.

[69] Min-Shiang Hwang, Cheng-Chi LEE, and Yan-Chi LAI. Traceability on low-computation partially blind signatures for electronic cash. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, E85-A:1181–1182, 2002. `http://ci.nii.ac.jp/naid/110003209141/en/`.

[70] Min-Shiang Hwang, Cheng-Chi Lee, and Yan-Chi Lai. An untraceable blind signature scheme. *IEICE Trans Fundam Electron Commun Comput Sci (Inst Electron Inf Commun Eng)*, E86-A(7):1902–1906, 2003.

[71] David Chaum. Blinding for unanticipated signatures. In *EUROCRYPT*, pages 227–233, 1987.

[72] Sebastiaan H. von Solms and David Naccache. On blind signatures and perfect crimes. *Computers & Security*, 11(6):581–583, 1992.

[73] Silvio Micali. Fair public-key cryptosystems. In *CRYPTO*, pages 113–138, 1993.

[74] Markus Stadler, Jean-Marc Piveteau, and Jan Camenisch. Fair blind signatures. In *EUROCRYPT*, pages 209–219, 1995.

[75] Min-Shiang Hwang, Cheng-Chi Lee, and Yan-Chi Lai. Traceability on stadler et al.'s fair blind signature scheme. *IEICE Trans Fundam Electron Commun Comput Sci (Inst Electron Inf Commun Eng)*, E86-A(2):513–514, 2003.

[76] Jean-Sébastien Coron, David Naccache, and Julien P. Stern. On the security of rsa padding. In *CRYPTO'99*, pages 1–18, 1999.

[77] Min-Shiang Hwang, Eric Jui-Lin Lu, and Yan-Chi Lai. Traceability of fan-chen-yeh blind signature scheme. Technical Report CYUT-IM-TR-2001-009, Aug 2001.

[78] Hung-Yu Chien, Jinn-Ke Jan, and Yuh-Min Tseng. Rsa-based partially blind signature with low computation. In *Eighth International Conference on Parallel and Distributed Systems, 2001 (ICPADS2001)*, pages 385–389, 2001.

[79] Min-Shiang Hwang, Cheng-Chi Lee, and Yan-Chi Lai. Traceability on rsa-based partially signature with low computation. *Applied Mathematics and Computation*, 145(2-3):465–468, 2003. `http://www.sciencedirect.com/science/article/B6TY8-47TFP68-B/2/678950f4bb850303c823039ac3ce056e`.

[80] C.-I. Fan and C.-L. Lei. Efficient blind signature scheme based on quadratic residues. *Electronics Letters*, 32(9):811–813, Apr 1996.

[81] C.-I. Fan and C.-L. Lei. Low-computation partially blind signatures for electronic cash. *IEICE Transactions on Fundamentals*, E81-A:818–824, May 1998.

[82] Chun-I Fan and Chin-Laung Lei. User efficient blind signatures. *Electronics Letters*, 34(6):544–546, Mar 1998.

[83] C.-I. Fan and C.-L. Lei. Cryptanalysis on improved user efficient blind signatures. *Electronics Letters*, 37(10):630–631, May 2001.

[84] J. Camenisch, J. Piveteau, and M. Stadler. Blind signatures based on discrete logarithm problem. In *EUROCRYPT'94: Advances in Cryptology, Lecture Notes in Computer Science*, pages 428–432. Springer, 1994.

[85] NIST. Digital signature standard (dss). National Institute of Satndards and Technology, 1993. Technical Report, PUB XX, US Department Commerce.

[86] L. Harn. Cryptanalysis of the blind signatures based on the discrete logarithm problem. *Electronics Letters*, 31(14):1136, Jul 1995.

[87] P. Horster, M. Michels, and H. Petersen. Cryptanalysis of the blind signatures based on the discrete logarithm problem. *Electronics Letters*, 31(21):1827, Oct 1995.

[88] E. Mohammed, A.E. Emarah, and K. El-Shennawy. A blind signature scheme based on elgamal signature. In *EUROCOMM 2000. Information Systems for Enhanced Public Safety and Security. IEEE/AFCEA*, pages 51–53, 2000.

[89] Min-Shiang Hwang, Yuan-Liang Tang, and Yan-Chi Lai. Comment on :a blind signature scheme based on elgamal signature. Technical Report CYUT-IM-TR-2001-009, Aug 2001.

[90] Cheng-Chi Lee, Wei-Pang Yang, and Min-Shiang Hwang. Untraceable blind signature schemes based on discrete logarithm problem. *Fundamentae Informatica*, 55(3-4):307–320, 2003.

[91] Lin-Chuan Wu. Analysis of traceability attack on camenisch et al.'s blind signature schemes. In *ASIACCS*, page 366, 2006.

[92] Zuowen Tan. Improvement on a generalized scheme of proxy signature based on elliptic curves. In *CIS*, pages 677–681, 2007.

[93] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, pages 770–772, 1981.

[94] Neil Haller. The s/key one-time password system. In *Proceedings of the Internet Society Symposium on Network and Distributed Systems*, pages 151–157, 1994.

[95] All J. Atkinson, Daniel L. McDonald, Daniel L. Mcdonald, Randall J. Atkinson, Craig Metz, and Craig Metz. One time passwords in everything (opie): Experiences with building and using stronger authentication. In *Proc. of 5th USENIX Security Symposium*, pages 177–86, 1995.

[96] Chris J. Mitchell and Liqun Chen. Comments on the s/key user authentication scheme. *SIGOPS Oper. Syst. Rev.*, 30(4):12–16, 1996.

[97] Akihiro SHIMIZU, Tsutomu HORIOKA, and Hirohito INAGAKI. A password authentication method for contents communications on the internet. *IEICE transactions on communications*, 81(8):1666–1673, 1998. `http://ci.nii.ac.jp/naid/110003218235/en/`.

[98] Mohammad Peyravian and Nevenko Zunic. Methods for protecting password transmission. *Computers & Security*, 19(5):466–469, 2000.

[99] Cheng-Chi Lee, Li-Hua Li, and Min-Shiang Hwang. A remote user authentication scheme using hash functions. *SIGOPS Oper. Syst. Rev.*, 36(4):23–29, 2002.

[100] Ji-Hye Park, Seong Jun Yim, and Jik Hyun Chang. A secure remote user authentication scheme. In *Proceedings of the Third International Conference on Convergence and Hybrid Information Technology, 2008. ICCIT '08*, volume 2, pages 368 –373, Nov 2008.

[101] C.C. Chang and C.S. Laih. Remote password authentication with smart cards. *IEEE Proceedings on Computers and Digital Techniques*, 139(4):372, Jul 1992.

[102] W. Yang and S. Shieh. Password authentication schemes with smart cards. *Computers and Security*, 18(8):727–7062, 1999.

[103] C. K. Chan and L. M. Cheng. Cryptanalysis of a timestamp-based password authentication scheme. *Computers and Security*, 21(1):74–76, 2002.

[104] Hung-Min Sun. An efficient remote use authentication scheme using smart cards. *IEEE Transactions on Consumer Electronics*, 46(4):958–961, Nov 2000.

[105] Min-Shiang Hwang and Li-Hua Li. A new remote user authentication scheme using smart cards. *IEEE Transactions on Consumer Electronics*, 46(1):28–30, Feb 2000.

[106] T.C. Wu. Remote login authentication scheme based on a geometric approach. *Computer Communications*, pages 12–18, 1995.

[107] M .S. Hwang. Cryptanalysis of a remote login authentication scheme. *Computer Communications*, pages 742–744, 1999.

[108] David Paul Maher. Crypto backup and key escrow. *Commun. ACM*, 39(3):48–53, 1996.

[109] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: the three party case. In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 57–66, New York, NY, USA, 1995. ACM.

[110] Shengbao Wang, Zhenfu Cao, M.A. Strangio, and Lihua Wang. Cryptanalysis and improvement of an elliptic curve diffie-hellman key agreement protocol. *Communications Letters, IEEE*, 12(2):149–151, Feb 2008.

[111] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Commun. ACM*, 24(8):533–536, 1981.

[112] Victor S. Miller. *Uses of Elliptic Curve in Cryptography*, volume 218 of *Lectures notes on Computer Sciences*, chapter Advances in Cryptography-Proceedings of Crypto85, pages 417–426. Springer-Verlag, New York, USA, 1986.

[113] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[114] Neal Koblitz. *CM-Curves with Good Cryptographic Properties*, volume 576 of *Lectures notes on Computer Sciences*, chapter Advances in Cryptology-Proceedings of Crypto91, pages 279–287. Springer-Verlag, London, UK, 1991.

[115] Anoop MS. Elliptic curve cryptography - an implementation guide, January 2007. `http://hosteddocs.ittoolbox.com/AN1.5.07.pdf`.

[116] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 2nd edition, 2002.

[117] Certicom. Standards for efficient cryptography, sec 1: Recommended elliptic curve domain parameters, version 1.0, September 2000. `http://www.secg.org/download/aid-385/sec1\_final.pdf`.

[118] Certicom. Standards for efficient cryptography, sec 2: Recommended elliptic curve domain parameters, version 1.0, September 2000. `http://www.secg.org/download/aid-386/sec2\_final.pdf`.

[119] Zulfikar Amin Ramzan. Group blind digital signatures: Theory and applications. Master's thesis, Massachusetts Institute of Technology, May 1999.

[120] Peter Wayner. *Digital cash: Commerce on the Net*. Academic Press Professional Inc., San Diego, CA, USA, 1995.

[121] Laurie Law, Susan Sabett, and Jerry Solinas. How to make a mint: the cryptography of anonymous electronic cash. National Security Agency, Office of Information Security Research and Technology, Cryptology Division, June 1996.

[122] Bruce Schneier. *Applied cryptography: Protocols, Algorithms, and Source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1995.

[123] Zhengjun Cao and Lihua Liu. A strong rsa signature scheme and its application. In *SNPD '07: Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pages 111–115, Washington, DC, USA, 2007. IEEE Computer Society.

[124] NSA. The case for elliptic curve cryptography. National Security Agency- Central Security Service. `http://www.nsa.gov/business/programs/elliptic\_curve.shtml`.

[125] L. Lopez-Garcia, L. Martinez-Ramos, and F. Rodriguez-Henriquez. A comparative performance analysis of several blind signature schemes. In *5th International Conference on Electrical Engineering, Computing Science and Automatic Control, 2008. CCE 2008*, pages 310–315, Nov. 2008.

Dissemination of Work

Journals

1. **Debasish Jena** and Sanjay Kumar Jena. A Novel and Efficient Cryptosystem for Large Message Encryption. *International Journal of Information and Communication Technology, Inderscience Publishers*, 2010. To be published in Vol. 3, Issue 1.

2. Sairam Kulkarni, **Debasish Jena**, and Sanjay Kumar Jena. A Novel Secure Key Agreement Protocol using Trusted Third Party. *International Journal of Computer Science and Security*, 1(1):11–18, 2007.

3. **Debasish Jena** and Sanjay Kumar Jena. A Novel untraceable blind signature based on elliptic curve discrete logarithm problem. *International Journal of Computer Science and Network Security*, 7(6):269–275, 2007.

Conferences

1. **Debasish Jena**, Saroj Kumar Panigrahy, and Sanjay Kumar Jena. A Novel and Efficient Cryptosystem for Long Message Encryption. In *4th IEEE International Conference on Industrial and Information Systems 2009. ICIIS-09*. University of Peradeniya, Sri Lanka.

2. **Debasish Jena**, Saroj Kumar Panigrahy, and Sanjay Kumar Jena. A Novel Remote User Authentication Scheme using Smart Card based on ECDLP. In *4th IEEE International Conference on Industrial and Information Systems 2009. ICIIS-09*. University of Peradeniya, Sri Lanka.

3. **Debasish Jena**, Sanjay Kumar Jena. A Novel Visual Cryptography Scheme. In *IEEE International Conference on Advanced Computer Control, 2009. ICACC '09*, pages 207–211, Singapore, Jan. 2009.

4. **Debasish Jena**, Sanjay Kumar Jena, Debasisha Mohanty, and Saroj Kumar Panigrahy. A Novel Remote User Authentication Scheme using Smart Card

based on ECDLP. In *IEEE International Conference on Advanced Computer Control, 2009. ICACC '09*, pages 246–249, Singapore, Jan. 2009.

5. **Debasish Jena**, Saroj Kumar Panigrahy, Pradip Kumar Biswal, and Sanjay Kumar Jena. A Novel Protocol for Smart Card using ECDLP. In *First IEEE International Conference on Emerging Trends in Engineering and Technology, 2008. ICETET'08*, pages 838–843, Nagpur, India, July 2008.

6. **Debasish Jena**, Sanjay Kumar Jena, and B. Majhi. A Novel Blind Signature Scheme based on Nyberg-Rueppel Signature Scheme and Applying in Off-line Digital Cash. In *10th IEEE International Conference on Information Technology, (ICIT 2007)*, pages 17–20, NIT Rourkela, India, Aug. 2007.

Book Chapter

1. **Debasish Jena**, Sanjay Kumar Jena, Banshidhar Majhi, and Saroj Kumar Panigrahy *A Novel ECDLP based Blind Signature Scheme with an Illustration*, pages 59–68. Narosa Publishing House, 2008.

Debasish Jena

Assistant Professor
IIIT Bhubaneswar
Gothapatna, Malipada,
Bhubaneswar - 751003, Odisha, India

Ph: +91-674-3016011 (O), +91-9437230284 (M)

e-mail: debasish@iiit-bh.ac.in

Date of Birth

December 18, 1968

Permanent Address

Plot# 1112, Laxmi Sagar, Talasahi,
Bhubaneswar – 751 006, Odisha, India.

Qualification

- Ph.D. (Continuing)
  NIT Rourkela
- M.Tech. (CS)
  Utkal Universuty, Orissa [First Division with Honours]
- M.B.A (System & Finance)
  Utkal Universuty, Orissa [First Division]
- B.E. (CSE)
  Gulbarga University, Karnataka, [First division]
- +2 (Science)
  Council of Higher Secondary Education, Orissa, [First division]
- 10th
  Board of Secondary Education, Orissa, [First division]

Publications

- 04 Journal Articles
- 15 Conference Papers
- 03 Book Chapters